UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

CONFECTION DE ROUTES DE VÉHICULES POUR LE TRANSPORT DE PLUSIEURS ORIGINES À PLUSIEURS DESTINATIONS

par

Yvan Dumas

Mémoire de M.Sc., département d'informatique et de recherche opérationnelle, dirigé par Jacques Desrosiers

Université de Montréal
Centre de recherche sur les transports - Publication #434
Juillet 1985

Dépôt légal - 4e trimestre 1985

TABLE DES MATIERES

	Page
SOMMAIRE	vi
CHAPITRE 1 : INTRODUCTION	1
1.1 - Description du problème	3
1.2 - Formulation mathématique	5
CHAPITRE 2 : APPROCHES DE RESOLUTION	10
2.1 - Revue de la littérature	11
2.2 - Notre approche de résolution	15
2.2.1 - Génération de colonnes	17
CHAPITRE 3 : METHODE DE RESOLUTION D'UN SOUS-PROBLEME ...	21
3.1 - Formalisation de l'approche	22
3.2 - Réduction du nombre d'états	24
3.3 - Approches heuristiques	27
3.3.1 - Relaxation de la contrainte d'élimination de cycle	27
3.3.2 - Restriction sur h^1_j	29
3.3.3 - Restriction du nombre d'états par dominance augmentée	30
3.4 - Génération de plusieurs colonnes à la fois	31
CHAPITRE 4 : RESOLUTION DU PROBLEME MAITRE	34
4.1 - Etape 1 : résolution du problème maître restreint .	35
4.2 - Etape 2 : résolution des sous-problèmes	36
4.3 - Construction d'un bon problème maître restreint ...	39
CHAPITRE 5 : RESOLUTION DU PROBLEME MAITRE EN NOMBRES ENTIERS	41
5.1 - Coupe sur le nombre de véhicules	42
5.2 - Séparation et évaluation progressive	43

CHAPITRE 6 : IMPLANTATION 46
6.1 - L'algorithme de programmation dynamique 47
6.2 - Structure de données principale 51
6.3 - Construction et réduction du réseau 54
6.3.1 - Critères d'élimination des arcs 54
6.3.2 - Informations conservées à chaque arc 56

CHAPITRE 7 : RESULTATS 58
7.1 - Description des problèmes 59
7.1.1 - Demandes 59
7.1.2 - Véhicules 63
7.2 - Description des résultats 66
7.3 - Analyse des résultats 71
7.3.1 - Performance globale 71
7.3.2 - Problèmes réels 71
7.3.3 - Problèmes à demandes aléatoires (réseau réel) 73
7.3.4 - Problèmes complètement aléatoires 74
7.3.5 - Problèmes avec horaire sur les demandes ... 75
7.3.6 - Problèmes avec plus grande disponibilité des chaufeurs 76
7.3.7 - Problèmes avec plusieurs dépôts 77
7.3.8 - Restriction sur la charge 77
7.3.9 - Synthèse des résultats 79
7.4 - Comportement de l'algorithme 80
7.5 - Effets de différentes stratégies 89

CHAPITRE 8 : CONCLUSION 91

REMERCIEMENTS 94

BIBLIOGRAPHIE 95

TABLE DES ILLUSTRATIONS

- FIGURES -

Page

CHAPITRE 3

Figure 3.1 : Graphique sur la dominance 26
Figure 3.2 : Graphique sur la dominance augmentée 30

CHAPITRE 5

Figure 5.1 : L'arbre d'énumération implicite 44

CHAPITRE 6

Figure 6.1 : Structure de l'arbre des états 52

CHAPITRE 7

Figure 7.1 : Graphique du temps d'exécution vs les étapes 82
Figure 7.2 : Graphique du nombre de colonnes générées vs les étapes 83
Figure 7.3 : Graphique du nombre de demandes traitées vs la colonne générée 86
Figure 7.4 : Graphique du temps d'exécution vs la colonne générée 87
Figure 7.5 : Graphique du nombre maximum d'états conservés vs la colonne générée 88

TABLE DES ILLUSTRATIONS

- TABLEAUX -

Page

CHAPITRE 6

Tableau 6.1 : Critères d'élimination des arcs 55
Tableau 6.2 : Informations sur chaque arc 57

CHAPITRE 7

Tableau 7.1 : Demandes de base 60
Tableau 7.2 : Variantes au groupe de demandes D1 63
Tableau 7.3 : Véhicules 65
Tableau 7.4 : Problèmes de base 68
Tableau 7.5 : Variantes de D1.V1 (horaire) 69
Tableau 7.6 : Variantes de D1.V1 (dépôt et charge) 70
Tableau 7.7 : Compilation sur les étapes majeures 84
Tableau 7.8 : Comparaison des différentes stratégies pour Z(pl) 89

SOMMAIRE

Nous considérons le problème de confection de routes et d'horaire pour une flotte mixte de véhicules devant effectuer un ensemble de demandes de transport. Chaque demande consiste à transporter une quantité de marchandise d'un lieu d'origine à un lieu de destination, avec une contrainte d'intervalle de temps sur l'heure de visite de chacun des lieux. A la flotte mixte de véhicules peuvent être associés plusieurs dépôts. Nous voulons résoudre ce problème de façon à minimiser les coûts de transport tout en respectant la capacité, la disponibilité et l'horaire des véhicules. Ce problème est en fait une généralisation du voyageur de commerce multiple où plusieurs dépôts sont considérés et des contraintes de capacité, d'horaire, de précédence et de pairage sont rajoutées. Il réfère à la classe de problème "Many-To-Many Routing and Scheduling Problem" (Bodin et al. [4]).

Nous présentons une méthode optimale de résolution, dans le cas où les contraintes du problème sont restrictives, en utilisant une approche de recouvrement. Le recouvrement des demandes avec routes réalisables est résolu par génération de colonnes. Chacune des routes réalisables est définie par un problème de plus court chemin avec contraintes additionnelles et est générée à partir d'un algorithme de programmation dynamique.

Quelques idées sont également données afin d'obtenir une excellente méthode heuristique permettant de résoudre de plus gros problèmes.

Des résultats numériques sont présentés pour des problèmes ayant jusqu'à 55 demandes.

CHAPITRE 1

INTRODUCTION

1. INTRODUCTION

Depuis longtemps, le transport est devenu un service vital et coûteux. Qu'il s'agisse du transport de personnes, de marchandise ou autre, par des véhicules (camions, avions, trains, etc), dans chaque contexte on doit résoudre le problème logistique de la minimisation des coûts d'opérations du transport. Ce problème logistique est difficile à résoudre vu les nombreuses possibilités de décisions au niveau de la confection des routes et/ou horaires des véhicules afin de satisfaire la demande des clients.

Il existe un grand nombre de situations pratiques pour lesquelles le problème logistique se distingue tant au niveau des contraintes qu'au niveau de l'objectif. Bodin et al. [4] classifie les problèmes de confection de routes et d'horaires en trois grandes classes. La première classe est celle des problèmes de confection de routes de véhicules ("vehicle routing problem"), notée VRP. Le VRP demande à trouver un ensemble de routes de véhicules de capacité limitée (et parfois de déplacement limité) afin de desservir un ensemble de demandes à coût de transport minimum. Chaque demande consiste en la visite d'un seul lieu et aucune dimension temporelle n'est considérée.

La seconde classe est celle des problèmes de confection d'horaires de véhicules ("vehicle scheduling problem"), notée VSP. Le VSP est un VRP où en plus, chaque demande a une heure de service fixe qui doit être respectée et dont les véhicules sont sujets à un horaire.

Finalement, la dernière classe est celle des problèmes de confection de routes et d'horaire de véhicules ("vehicle routing and scheduling problems"), notée VRSP. Le VRSP est un VRP où en plus, chaque demande consiste en la visite d'un ou 2 lieux par un même véhicule (prise et/ou dépôt) et dont chaque lieu doit être visité dans un intervalle de temps.

Cette terminologie n'est pas commune à tous les auteurs et Christofides [6] en mentionne certaines autres.

1.1 Description du problème

La classe de problème étudiée dans ce document est celle du VRSP. Les deux autres classes sont des cas particuliers du VRSP. Les principales caractéristiques que nous allons traiter sont les suivantes:

A chaque client (demande) sont associés:

C1 - Une quantité de marchandise à transporter;
C2 - Un temps de manutention de l'entité à transporter;
C3 - Un lieu de chargement et un lieu de déchargement (contrainte de précédence);
C4 - Des intervalles de temps dans lesquels le chargement et le déchargement doivent s'effectuer (contraintes d'horaire);

A chaque véhicule sont associés:

V1 - Une capacité de chargement;
V2 - Un lieu de départ et un lieu de retour (ils peuvent être distincts);
V3 - Un horaire de disponibilité;
V4 - Un coût fixe d'utilisation, une vitesse de déplacement, un coût de déplacement à vide et un coût de déplacement avec charge.

Les véhicules d'une flotte sont regroupés par type de véhicules où les véhicules de même type ont les mêmes caractéristiques V1 à V4.

A chaque type de véhicules sont alors associés:

T1 - Le nombre de ces véhicules;
T2 - Les caractéristiques V1 à V4.

Deux objectifs principaux peuvent être considérés:

O1 - Confectionner les routes des véhicules disponibles de façon à minimiser le déplacement et à satisfaire la demande des clients;
O2 - Confectionner les routes des véhicules disponibles de façon à minimiser le nombre de véhicules utilisés et leur déplacement et de façon à satisfaire la demande des clients.

En pratique, il existe dans certaines situations plusieurs autres caractéristiques pouvant être considérées (Christofides [6]); mais la situation traitée possède plusieurs aplications réelles (Guinet [13], Desrosiers et al. [10] et Bodin et al. [4]).

A notre connaisance tous les algorithmes disponibles pour résoudre le type de problème énoncé sont des heuristiques. Nous proposons dans ce document, une formulation du problème et une méthode de résolution exacte qui est efficace lorsque les contraintes sont restrictives i.e. lorsque chaque véhicule dessert peu de demandes. Le problème sera appliqué dans le cadre du transport de marchandise.

1.2 Formulation mathématique

On définit les **ensembles** suivants servant à construire un réseau:

$D = \{1,2,\ldots,n\}$: l'ensemble des demandes,

$P^+ = \{j^+ \mid j \in D\}$: l'ensemble des noeuds d'origine des demandes,

$P^- = \{j^- \mid j \in D\}$: l'ensemble des noeuds de destination des demandes,

$P = P^+ \cup P^-$: l'ensemble des noeuds associés aux demandes,

$V = \{n+1,\ldots,n+q\}$: l'ensemble des types de véhicules,

$G^+ = \{(n+j)^+ \mid (n+j) \in V\}$: l'ensemble des noeuds d'origine des types de véhicules,

$G^- = \{(n+j)^- \mid (n+j) \in V\}$: l'ensemble des noeuds de destination des types de véhicules.

A chaque **demande** i sont associés:

i^+ : le noeud correspondant au lieu d'origine de la demande,

i^- : le noeud correspondant au lieu de destination de la demande,

$[a_i+, b_i+]$: un intervalle de temps à l'intérieur duquel l'origine doit être visitée,

$[a_i-, b_i-]$: un intervalle de temps à l'intérieur duquel la destination doit être visitée,

d_i : la quantité de marchandise à transporter,

M_i+, M_i- : les temps de manutention de la marchandise à l'origine et à la destination.

A chaque type g de **véhicules** sont associés:

g^+ : le noeud correspondant au lieu de départ,
g^- : le noeud correspondant au lieu de retour,
$[a_g^+, b_g^+]$: un intervalle de temps pendant lequel le véhicule (ou chauffeur) est disponible,
$[a_g^-, b_g^-]$: idem,
C^g : la capacité de ces véhicules,
Q^g : le nombre de véhicules de ce type,
W^g : $\{1,\ldots,Q^g\}$ l'ensemble de ces véhicules,
d_g : vaut 0,
M_g^+, M_g^- : valent 0.

La formulation inclue les **variables** suivantes:

x_{ij}^g : variable de flot, prenant la valeur 1 si un véhicule de type g utilise l'arc (i,j) et 0 sinon,
t_i : variable de temps, représentant l'heure de passage au noeud i,
y_i^g : variable de chargement, représentant la quantité de marchandise présente dans un véhicule de type g à la sortie du noeud i.

Du noeud i au noeud j, on note respectivement par t_{ij} et $c_{ij}^g(y_i^g)$ le temps de déplacement et la fonction de coût de déplacement du véhicule g. La fonction de coût $c_{ij}^g(y_i^g)$ dépend de la variable y_i^g et inclue un coût fixe d'utilisation du véhicule si $i \in G^+$. Elle prend la forme suivante:

$$c_{ij}^g(y_i^g) = \begin{cases} r + t_{ij} \cdot \alpha & \text{si } i \in G^+ \\ t_{ij} \cdot \beta & \text{si } y_i^g > 0 \quad i \notin G^+ \\ t_{ij} \cdot \alpha & \text{si } y_i^g = 0 \quad i \notin G^+ \end{cases}$$

où r est le coût fixe d'utilisation du véhicule, α et β sont des constantes de pondération pour le coût de déplacement du véhicule à vide et avec charge respectivement.

Finalement on définit l'ensemble $I = \{(i,j) \mid i,j \in P \cup G^+ \cup G^-\}$ comme étant celui des arcs du réseau qui satisfont les contraintes d'horaires, de capacité, etc. Nous verrons dans le chapitre sur l'implantation quels sont les arcs qui ne sont pas conservés.

Notre problème de transport se formule comme suit, dans le cas où $Q^g=1$ pour $g \in V$ i.e. on considère tous les véhicules distincts:

(1) $$\text{MIN } Z = \sum_{g \in V} \sum_{(i,j) \in I} c^g_{ij}(y^g_i) \cdot x^g_{ij}$$

sujet à

(2) $$\sum_{g \in V} \sum_{i \mid (i,j) \in I} x^g_{ij} = 1 \qquad j \in P^+$$

(3) $$\sum_{i \mid (i,j) \in I} x^g_{ij} - \sum_{i \mid (j,i) \in I} x^g_{ji} = 0 \qquad j \in P,\ g \in V$$

(4) $$\sum_{j \mid (g^+,j) \in I} x^g_{g^+j} \leq Q^g \qquad g \in V$$

(5) $$\sum_{j \mid (h,j) \in I} x^g_{hj} = \sum_{i \mid (i,k) \in I} x^g_{ik} = 0 \qquad h \in G^+,\ h \neq g^+,\ k \in G^-,\ k \neq g^-,\ g \in V$$

(6) x^g_{ij} binaire $\qquad (i,j) \in I,\ g \in V$

(7) $$\sum_{i \mid (j^+,i) \in I} x^g_{j^+i} = \sum_{i \mid (i,j^-) \in I} x^g_{ij^-} \qquad j \in D,\ g \in V$$

(8) $x^g_{ij} = 1 \implies y^g_i + d_j = y^g_j \qquad j \in P^+,\ (i,j) \in I,\ g \in V$

(9) $x^g_{ij} = 1 \implies y^g_i - d_j = y^g_j \qquad j \in P^-,\ (i,j) \in I,\ g \in V$

(10) $x^g_{ij} = 1 \implies t_i + t_{ij} + M_i \leq t_j \qquad (i,j) \in I,\ g \in V$

(11) $t_i^+ < t_i^- \qquad i \in D$

(12) $0 \leq y^g_j \leq C^g \qquad j \in D,\ g \in V$

(13) $a_i \leq t_i \leq b_i \qquad i \in P \cup G^+ \cup G^-$

Dans le cas où $Q^g > 1$ pour au moins 1 type de véhicules, cette formulation n'est plus valide à cause des contraintes de pairage (7). Il faut alors se définir une nouvelle variable x_{ij}^{gk} comme suit:

x_{ij}^{gk} : variable de flot, prenant la valeur 1 si le véhicule k de type g utilise l'arc (i,j) et 0 sinon,

et remplacer dans les équations (1) à (5) et (8) à (10) x_{ij}^{g} par $\sum_{k \in W^g} x_{ij}^{gk}$ et remplacer dans les équations (6) et (7) x_{ij}^{g} par x_{ij}^{gk} et remplacer $g \in V$ par $g \in V$, $k \in W^g$. Cette nouvelle formulation a pour avantage que le nombre de contraintes (3) à (5) et (8) à (10) soit fonction du nombre de types de véhicules et non du nombre de véhicules.

L'objectif (1) consiste à mimimiser le nombre de véhicules utilisés (ce qui est réalisé par l'ajout d'un coût fixe dans la fonction de coût de déplacement) et le déplacement de ces véhicules. Les contraintes (2) et (6) spécifient que chaque noeud d'origine, doit être visité une seule fois par exactement un véhicule. Notons que cette exigeance est redondante pour les noeuds de destination lorsque les contraintes (2) et (7) sont présentes. Les contraintes (3) et (4) décrivent respectivement la conservation du flot à chaque noeud et la limite d'utilisation des véhicules pour chaque type. Les contraintes (5) correspondent aux dépôts, chaque type de véhicules n'ayant pas accès aux dépôts ne lui étant pas associés. Les contraintes (7) spécifient que la marchandise de chaque demande est prise et livrée par le même véhicule (contraintes de pairage). Les relations (8) à (10) décrivent les exigences de compatibilité entre les routes, la charge et l'horaire des véhicules. Et finalement les contraintes (11), (12) et (13) imposent respectivement la relation de précédence entre les noeuds d'une même demande, la limite de capacité par type de véhicules et les intervalles de temps à chaque noeud.

Nous pouvons remarquer que la taille du problème est fonction du nombre de demandes (|D|), du nombre de types de véhicules (|V|), du nombre de véhicule et du nombre d'arcs du réseau (|I|).

Le problème sans les contraintes (4) possède une solution réalisable. En effet, il suffit d'envoyer un véhicule par demande pour avoir une solution réalisable. Par contre dans notre cas, nous ne sommes plus assurés de la réalisabilité du problème car les contraintes (4) nous limitent sur l'utilisation des véhicules.

Notre problème est une généralisation du voyageur de commerce multiple où des contraintes de capacités, d'horaires, de précédences et de pairage ont été rajoutées. Remarquons toutefois que les relations (8) à (10) ne sont pas des contraintes linéaires. Desrosiers, Soumis et Desrochers [9] ont montré qu'en linéarisant les relations (10), celles-ci généralisent les contraintes d'élimination de sous-tours proposées par Miller, Tucker et Zemlin [16] pour le problème du voyageur de commerce: en effet, les temps de visite aux noeuds étant strictement croissants, il ne peut y avoir de sous-tours.

CHAPITRE 2

APPROCHES DE RESOLUTION

2. APPROCHES DE RESOLUTION

Ce chapitre présente une revue de la littérature sur des approches de résolution du VRSP et du VRP et une nouvelle approche de résolution pour le type de problème énoncé au chapitre 1.

2.1 Revue de la littérature

Il existe dans la littérature très peu d'articles concernant le VRSP comparativement au VRP (Bodin et al. [4]). C'est pourquoi cette section portera sur quelques récents travaux concernant le VRP et présentant des approches exactes et quelques autres travaux concernant le VRSP et présentant des approches heuristiques et exactes de résolution. Pour un bon survol des approches de résolution du VRP on peut consulter [3,4,12] et [4,10,20] pour le VRSP.

Christofides (1976) [6] classifie et discute de méthodes de résolution exactes et heuristiques pour différents problèmes de la classe du VRSP et du VRP. Dans la classe des méthodes exactes, on retrouve les méthodes basées sur l'énumération des routes d'un véhicule ("set partionning algorithm") et la méthode basée sur la génération des routes réalisables (non pas sur la génération de toutes les routes à priori, mais bien au besoin) à partir d'un arbre d'énumération développé par l'auteur. Ces méthodes présentées peuvent tenir compte d'un grand nombre de contraintes de différents types sur les routes et sont plus efficaces lorsque le nombre de contraintes restrictives est grand. La méthode developpée par l'auteur a permis de résoudre des problèmes de 31 clients pour le VRSP.

Plus récemment Christofides, Mingozzi et Toth (1981) [7] ont développé trois nouveaux algorithmes optimaux pour le VRP dans la situation où les véhicules sont identiques et les contraintes de capacité et de distance totale permise sur la route sont restrictives. Les algorithmes développés sont de type séparation et évaluation progressive ("branch and bound") et sont basés sur le calcul de deux bornes inférieures différentes obtenues des sous-problèmes de q-chemins ou de m-arbres qui sont résolus en nombres entiers. Cette approche a permis de traiter de façon optimale des problèmes jusqu'à 25 clients et les bornes obtenues sont au moins à 93 % du coût optimal.

La résolution de sous-problèmes en nombres entiers a aussi été utilisée dans une approche de génération de colonnes. Des routes respectant les contraintes sont générées en résolvant des sous-problèmes en nombres entiers. Par la suite, on choisit un sous-ensemble de routes de coût minimum visitant chaque client en résolvant un problème de recouvrement. Cette approche a permis à Balinski et Quandt [1] de résoudre de petits problèmes en générant toutes les routes réalisables. Rao et Zionts [18] ont proposé de traiter de plus grands problèmes en procédant par génération de colonnes mais n'ont pas réalisé leur suggestion.

Christofides, Mingozzi et Toth (1981) [8] proposent aussi, une procédure générale de relaxation dans laquelle les états associés à une récursion de programmation dynamique donnée sont relaxés. La solution de la récursion relaxée fournit une borne inférieure laquelle peut être utilisée dans un arbre de séparation et d'évaluation progressive (noté SEP) pour la résolution du problème. Cette méthode de relaxation d'états est analogue à la relaxation Lagrangienne en programmation en nombres entiers. Les auteurs font également un survol de cette nouvelle méthodologie en plus d'étudier, entre autres, le VRP.

Magnanti (1981) [15] présente un document de synthèse sur l'optimisation combinatoire et la gestion de flotte de véhicules. Différents modèles de formulation du VRP ("set covering, vehicle flow, commodity flow") sont présentés et discutés ainsi que des approches de résolution reliées aux formulations. Il discute également du VRSP de façon générale avec différentes perspectives d'approches de résolution dont les méthodes combinatoires, les méthodes de flot dans un réseau et de programmation linéaire, les méthodes de SEP, de programmation dynamique et de décomposition en programmation en nombres entiers.

Nobert (1982) [17] dans sa thèse de doctorat évalue plusieurs variantes d'algorithmes utilisant des coupes pour différentes extentions du voyageur de commerce multiple, dont le VRP. Pour l'extension au VRP, en plus de relaxer les contraintes d'intégralité, l'approche relaxe les contraintes d'élimination des sous-tours illégaux (incluant implicitement les contraintes de capacité). L'intégralité est obtenue au moyen d'une technique de SEP; les contraintes d'élimination de sous-tours sont introduites seulement lorsque nécessaires. Cette approche peut résoudre des VRP, dont les véhicules sont identiques, jusqu'à 30 clients. Elle est plus efficace pour les problèmes où les contraintes de capacité sont peu restrictives et ne peut résoudre des problèmes où les contraintes de capacité sont très restrictives. Laporte et Nobert (1983) [14] utilisent sensiblement le même algorithme que Nobert (1982) [17] pour résoudre le VRP avec des véhicules identiques. Trois variantes de l'algorithme de base sont discutées. ILs obtiennent des solutions optimales pour des problèmes de 15 à 50 villes.

Bodin et al. (1983) [4] présentent une synthèse de l'état de l'art dans la confection de routes et d'horaires de

véhicules. La majorité des méthodes présentées pour le VRP et le VRSP sont heuristiques. Les auteurs font la remarque que les méthodes exactes disponibles n'ont pu résoudre que de petits problèmes et ne sont pas d'un grand attrait. Parmi les problèmes traités, deux d'entre eux se rapprochent de celui présenté dans ce document, soit (sous appelation anglaise): "tractor trailor routing and scheduling problem with partial loads" et "multi-vehicle dial-a-ride routing and scheduling problem with time windows". L'approche de résolution proposée pour le premier problème est une approche séquentielle à deux pas; la confection de routes suivie de la confection d'horaires. Celle proposée pour le deuxième problème est une approche de type regroupement en premier et confection de routes en second ("cluster first and routing second").

Desrosiers, Soumis et Desrochers (1983) [9] présentent une approche optimale pour un cas du VRSP qui considère un ensemble de parcours ayant une durée, un coût et un intervalle de temps pendant lequel ils doivent être effectués à partir d'une flotte de véhicules identiques. L'approche de résolution procède par génération de colonnes produites par un sous-problème de chemin le plus court avec contraintes de temps. Des problèmes jusqu'à 151 parcours sont résolus optimalement dans le contexte du transport scolaire où chaque route de véhicule contient peu de parcours.

Solomon (1983) [20] présente une étude et une analyse de modèles et d'algorithmes pour le VRSP sans contrainte de précédence. Des méthodes approximatives y sont présentées.

Desrosiers, Soumis et Dumas (1985) [10] présentent une méthode approximative pour résoudre sensiblement le même type de problème étudié dans ce document, dans le cadre du transport

pour personnes handicappées. Le problème est résolu par mini-regroupement en premier et confection de routes sur les mini-groupes en second ("mini-clustering first and optimal routing second"). Un algorithme approximatif de mini-regroupement groupe ensemble les personnes transportées en même temps dans le même véhicule. Le problème de confection de routes sur les mini-groupes est optimalement résolu en utilisant un algorithme de génération de colonnes. Des problèmes jusqu'à 200 clients sont facilement résolus.

Guinet (1984) [13] étudie également le même type de problème que nous. L'approche heuristique proposée est aussi de type mini-regroupement en premier et confection de routes sur les mini-groupes en second. La confection de routes est résolue comme un problème d'affectation avec réemploi sous contraintes.

Lorsque les contraintes du problème sont peu restrictives, la meilleure approche optimale jusqu'ici est celle de relaxation/réintroduction [14,17] de contraintes difficiles ou encombrantes. Par contre lorsque le problème est fortement contraints, l'approche résolvant des sous-problèmes est plus efficace [7,9,10]. C'est cette dernière approche qui est exploitée dans ce document. Elle en est une de recouvrement et le recouvrement des demandes avec routes réalisables est résolu par génération de colonnes.

2.2 Notre approche de résolution

La formulation présentée dans la section 1.2 est en terme de variables de flots. Nous reformulons dans cette section le problème en terme de routes réalisables. On utilise alors la notation suivante:

Ω_g^i : l'ensemble des routes des véhicules de type g satisfaisant l'horaire, la limite de capacité, les relations de précédence et de pairage et incluant le noeud i,

Ω_g : $\bigcup_{i \in P^+} \Omega_g^i$ l'ensemble des routes des véhicules de type g,

c_r^g : coût de la route r du véhicule de type g (somme des coûts de déplacement sur les arcs utilisés de I et du coût fixe d'utilisation du véhicule),

x_r^g : variable de décision prenant la valeur 1 si la route r d'un véhicule de type g est utilisée et 0 sinon.

La nouvelle formulation s'écrit comme un problème de partitionnement avec contraintes de disponibilité ("base constrained set partitioning problem" [19]). Elle se formule comme suit, où la correspondance en équation avec la formulation initiale (1) à (13) est indiquée:

(14)≈(1) MIN $\sum_{g \in V} \sum_{r \in \Omega_g} c_r^g \cdot x_r^g$

sujet à

(15)≈(2) $\sum_{g \in V} \sum_{r \in \Omega_g^i} x_r^g = 1 \qquad i \in P^+$

(16)≈(4) $\sum_{r \in \Omega_g^{g+}} x_r^g \leq Q^g \qquad g \in V$

(17)≈(6) $x_r^g = 0,1 \qquad r \in \Omega_g,\ g \in V$

Notons que pour les contraintes (15), il est encore suffisant que $i \in P^+$ plutôt que $i \in P$ puisque les routes satisfont les contraintes de pairage et que $\Omega_g^{i+} = \Omega_g^{i-}$.

Pour résoudre ce problème nous relaxons premièrement les contraintes (17) en les remplaçant par les contraintes:

(17') $x_r^g \geq 0 \qquad r \in \Omega_g, \; g \in V$

Ce problème devient linéaire et peut se résoudre par l'algorithme du simplexe. Une solution entière est obtenue dans un arbre d'énumération en fixant des portions de routes à être inclues dans la solution après avoir posé une coupe exigeant l'utilisation d'un nombre entier de véhicules.

A cause du grand nombre de variables dans cette nouvelle formulation, la technique de génération de colonnes est utilisée pour résoudre le problème relaxé. Nous appelerons par la suite "**problème maître**" ce problème relaxé.

2.2.1 Génération de colonnes

La méthode de résolution par génération de colonnes correspond à une décomposition de Dantzig-Wolfe. Desrosiers, Soumis et Desrochers [9] et Desrosiers, Soumis et Dumas [10] l'ont utilisée dans le contexte du transport scolaire et du transport des handicapés respectivement.

En termes de méthode de décomposition on appelera **problème maître restreint**, le problème maître avec seulement un sous-ensemble de variables et on appelera **sous-problème** le problème qui consiste à générer une route (colonne) réalisable, pour un type de véhicules, du problème maître. Nous avons alors $|V|$ sous-problèmes différents puisque le problème maître inclus $|V|$ types de véhicules. Ils correspondent à résoudre des problèmes de plus court chemin avec contraintes d'horaire, de capacité, de précédence et de pairage dont l'objectif est de trouver la route (colonne) de plus petit coût relatif (noté $\bar{c}_r^g$).

Les étapes de base de l'algorithme de la génération de colonnes pour résoudre le problème maître sont les suivantes:

Etape 0 : initialiser le problème maître restreint.
Etape 1 : résoudre le problème maître restreint à l'optimalité.
Etape 2 : trouver la route (colonne) de plus petit coût relatif sur le réseau.
Si ce coût relatif est positif alors la solution optimale du problème maître restreint est une solution optimale du problème maître et l'algorithme se termine, sinon on poursuit.
Etape 3 : rajouter la nouvelle colonne trouvée à l'étape 2 au problème maître restreint et reprendre l'étape 1.

La colonne de plus petit coût relatif est identifiée par le type de véhicules g' et la route réalisable r' tel que:

$$\bar{c}^{g'}_{r'} = \underset{r,g}{MIN}\ \bar{c}^{g}_{r} = \underset{g\in V}{MIN}\ \{\ \underset{r\in\Omega_g}{MIN}\ \bar{c}^{g}_{r}\ \}.$$

Ce qui revient à résoudre $|V|$ sous-problèmes. Formulons maintenant les sous-problèmes.

Soit les variables duales σ_i, $i\in P^+$ associées aux contraintes (15) et π_g, $g\in V$ associées aux contraintes (16). L'objectif d'un sous-problème $g\in V$ est :

$$\underset{r\in\Omega_g}{MIN}\ \bar{c}^{g}_{r} = \underset{r\in\Omega_g}{MIN}\ \{\ c^{g}_{r} - \sum_{\substack{i\in P^+ \\ i\,|\,r\in\Omega^{i}_{g}}} \sigma_i - \pi_g\ \}$$

$$\equiv \underset{(i,j)\in I}{MIN} \sum \{\ c^{g}_{ij}(y^{g}_{i})\cdot x^{g}_{ij} - \sigma_i\cdot x^{g}_{ij} - \pi_g\ \} \text{ où } \sigma_i=0,\ i\notin P^+.$$

Le sous-problème pour le type de véhicules $g \in V$ se formule comme suit:

$$(18) \qquad \text{MIN } Z = \sum_{(i,j)\in I} [\, c^g_{ij}(y^g_i) - \sigma_i \,] \cdot x^g_{ij} - \pi_g$$

sujet à

$$(19) \approx (3)+(5) \qquad \sum_{i\,|\,(i,j)\in I} x^g_{ij} - \sum_{i\,|\,(j,i)\in I} x^g_{ji} = \begin{cases} +1 & j = g^- \\ 0 & j \in P \\ -1 & j \in g^+ \end{cases}$$

$$(20) \approx (6) \qquad x^g_{ij} \text{ binaire} \qquad (i,j)\in I$$

$$(21) \approx (7) \qquad \sum_{i\,|\,(j^+,i)\in I} x^{g^+}_{ji} = \sum_{i\,|\,(i,j^-)\in I} x^g_{ij^-} \qquad j \in D$$

$$(22) \approx (8) \qquad x^g_{ij} = 1 \implies y^g_i + d_j = y^g_j \qquad (i,j)\in I$$

$$(23) \approx (9) \qquad x^g_{ij} = 1 \implies y^g_i - d_j = y^g_j \qquad (i,j)\in I$$

$$(24) \approx (10) \qquad x^g_{ij} = 1 \implies t_i + t_{ij} + M_i \le t_j \qquad (i,j)\in I$$

$$(25) \approx (11) \qquad t_{i^+} < t_{i^-} \qquad i \in D$$

$$(26) \approx (12) \qquad 0 \le y^g_j \le C^g \qquad j \in D$$

$$(27) \approx (13) \qquad a_i \le t_i \le b_i \qquad i \in P \cup \{g^+, g^-\}$$

La relation (19) représente l'équation de conservation de flot pour $j \in P$ (≈équation (3)) et la contrainte de dépôt (≈équation (5)). Nous avons spécifié un flot unitaire à la source (g^+) et au puits (g^-) afin de borner le sous-problème. Au point extrême obtenu, on fait correspondre le rayon extrême du sous-problème non-borné (voir [9]).

Finalement, le problème maître restreint est résolu par l'algorithme du simplexe et les sous-problèmes sont résolus par un algorithme de programmation dynamique. Notons que cette dernière technique nous permet d'obtenir des solutions optimales entières pour les sous-problèmes. Les méthodes de résolution des sous-problèmes, du problème maître et du problème maître en nombres entiers seront discutées plus en détail dans les chapitres 3, 4 et 5 respectivement.

CHAPITRE 3

METHODE DE RESOLUTION D'UN SOUS-PROBLEME

3.0 Méthode de résolution d'un sous-problème

Pour un type de véhicule donné, nous devons trouver le plus court chemin pour se rendre de son lieu de départ (noeud g^+) à son lieu de destination (noeud g^-) en satisfaisant les contraintes d'horaires, de capacité, de précédences et de pairages. La technique employée pour résoudre ce problème est la programmation dynamique. Cette technique permet d'obtenir des solutions entières et tire profit des nombreuses contraintes du problème qui limite la combinatoire.

3.1 Formalisation de l'approche

Nous savons qu'à la programmation dynamique sont reliées les notions d'étapes, d'états et de fonction récursive. Pour établir la méthode de résolution, formalisons ces notions en correspondance avec notre problème.

Notion d'état: On caractérise une route par un état (ou bilan de la route) défini par le triplet (S,j,t_j) où S est l'ensemble des noeuds visités, j le dernier noeud de cette route et t_j l'heure de passage à ce noeud. Le coût de cet état est défini par la fonction $f_k(S,j,t_j)$ où $k = |S|$.

Notion d'étape: A la première étape nous construisons les routes qui partent du dépôt et qui ont visité un noeud. A la $k^{ème}$ étape nous construisons les routes ayant visité k noeuds, à partir des états (ou routes) de la $(k-1)^{ème}$ étape. Le processus s'arrête lorsque nous ne pouvons plus construire de routes à une étape ou si k a atteint sa borne maximale (kmax).

Notion de fonction récursive: La fonction récursive nous permet de passer d'une étape à l'autre en nous assurant de l'optimalité.

Elle prend la forme suivante pour $S \subseteq P$:

$$(28)\quad f_k(S,j,t_j) = \underset{i\in h_j(S)}{\text{MIN}} \{ f_{k-1}(S-\{j\},i,t_i) + c_{ij}(C(S)) \mid \begin{array}{l} a_j \le t_j \le b_j, \\ t_j \ge t_i + t_{ij} + M_i, \\ C(S) \le C^g, \\ Pr(j) \subseteq S \end{array} \}$$

où $C(S) = \sum_{j^+\in S} d_j - \sum_{j^-\in S} d_j$, $\quad Pr(j) = \begin{cases} \{i^+\} \text{ si } j = i^-, i\in D \\ \emptyset \text{ sinon (i n'existe pas)} \end{cases}$

et $h_j(S) = \{i\in S \mid (i,j)\in I\}$.

La fonction C(S) fournit la quantité de marchandise dans le véhicule après avoir visité les noeuds de S, Pr(j) est un ensemble représentant l'éventuel noeud origine devant précéder j et h_j est l'ensemble des noeuds prédécesseurs possible de j étant donné S.

Dans la condition sur le minimum, nous retrouvons respectivement, les contraintes d'intervalle de temps, de compatibilité entre la route et l'horaire, de capacité et de précédence. Notons ici qu'on empêche tout cycle de se former puisque si j est le dernier noeud visité alors l'ensemble des noeuds visités avant j est S-{j}.

Cette formule de récursion est initialisée pour les origines des demandes par:

$$f_1(\{j\},j,t_j) = c_{g^+j}(0) \quad \text{tel que} \quad \begin{array}{l} a_j \le t_j \le b_j, \\ t_j \ge a_{g^+} + t_{g^+j}, \\ j\in P^+. \end{array}$$

La solution optimale sera donc une route se terminant en g^- (le noeud de retour du véhicule de type g) qui a engendré le plus court chemin de valeur $f_k^*(S^*,g^-,t^*)$ où

$$(29)\quad f_k^*(S^*,g^-,t^*) = \underset{\substack{(S,j,t_j)\in\Sigma \\ k=2,4,6,\ldots}}{\text{MIN}} \{ f_k(S,j,t_j) + c_{jg^-}(0) \mid \begin{array}{l} a_{g^-} \le t^* \le b_{g^-}, \\ t^* \ge t_j + t_{jg^-} + M_j, \\ C(S)=0 \end{array} \}$$

où Σ est l'ensemble des états réalisables de la récursion (28).

La condition supplémentaire que nous devons satisfaire dans cette deuxième relation est la contrainte de pairage (C(S)=0), i.e. avant de retourner au dépôt le véhicule doit avoir livré toute la marchandise prise en charge (si $i^+ \in S$ alors $i^- \in S$).

La récursion (28) est résolue de façon "forward" en commençant à appliquer la récursion pour k=1, ensuite nous passons à k=2 et ainsi de suite. De cette façon nous allons d'une étape à l'autre en ne considérant que les informations de l'étape précédente. Quant à la relation (29), elle est évaluée simultanément avec la relation (28), à chaque étape paire.

3.2 Réduction du nombre d'états

Il est bien connu que le principal inconvénient de la programmation dynamique réside dans le très grand nombre d'états générés et conservés. Même avec les contraintes incluses dans la récursion (28), le nombre d'états générés reste considérablement élevé. Il nous faut donc trouver des moyens pour réduire le nombre d'états générés et conservés.

Remarquons premièrement qu'il n'est pas nécessaire de conserver tous les états générés; seuls ceux de l'étape précédente sont nécessaires à la construction des états de l'étape suivante. On peut alors réduire l'espace mémoire requis en conservant les états de 2 étapes seulement. Nous pouvons même ne conserver que les états d'un peu plus d'une étape en résolvant la récursion (28) de façon ingénieuse (voir chapitre 6 sur l'implantation).

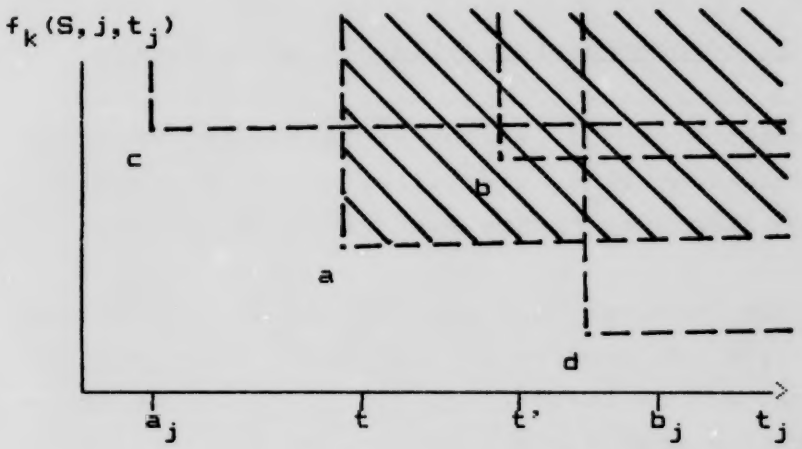
Une seconde technique employée par Desrosiers, Soumis et Dumas [10] afin de réduire davantage le nombre des états

conservés et générés est celle d'éliminer ceux qui ne peuvent permettre de construire des états réalisables dans les étapes suivantes. On dit que ces états sont non post-réalisables. Cette même technique a été employée dans notre algorithme. Idéalement elle consiste à vérifier si l'état représenté par (S,j,t_j) pourra engendrer l'état réalisable $(S \cup T, g^-, b_{g^-})$ qui nous intéresse dans la récursion (29) où $T = \{j^- \mid j^+ \in S \text{ et } j^- \notin S\}$ est l'ensemble des noeuds de destination non visités correspondant aux noeuds d'origine déjà visités. On veut en fait vérifier s'il est possible de livrer en temps toute la marchandise chargée dans le véhicule. Mais pour $|T| > 2$ cette vérification devient beaucoup trop coûteuse et nous devons nous restreindre à une vérification partielle en choisissant un sous-ensemble de T de cardinalité d'au plus 2. Nous n'avons alors que deux façons possibles d'ajouter 2 noeuds de destination à S avant d'aller au noeud g^-.

Une troisième réduction est encore possible puisque la récursion (28) produit beaucoup d'états redondants. Nous avons vu que cette récursion applique la dominance sur le coût en ne conservant, pour chaque état (S,j,t_j), que le coût de la meilleure route de cet état. Nous pouvons voir également qu'il est possible d'appliquer une dominance sur le temps, puisque pour deux états (S,j,t_j) et (S,j,t'_j) de même coûts et tel que $t_j < t'_j$ alors l'état (S,j,t'_j) est redondant dans la récursion (28). En d'autre mots, on dit qu'une route A domine une route B si cette dernière a visité les mêmes arrêts au même coût mais avec un plus grand temps de trajet. Graphiquement la surface de dominance sur le coût et le temps est représentée par le quadrant supérieur droit. Nous pouvons voir, à la figure 3.1, que l'état représenté par le point "a" domine celui représenté par le point "b" puisque ce dernier se trouve dans le quadrant supérieur droit du point "a". Finalement, nous dirons que l'état (S,j,t_j) domine l'état (S,j,t'_j) si (30) est satisfait.

$$(30) \quad \begin{cases} f_k(S,j,t'_j) - f_k(S,j,t_j) \geq 0 \\ \text{et} \quad t'_j - t_j \geq 0 \end{cases}$$

Figure 3.1 : Graphique sur la dominance



Cette dominance suplémentaire permet une réduction appréciable! Dans Desrosiers, Soumis et Desrochers [9] ces états dominant sont référés à des étiquettes dominantes.

Par les trois réductions précédentes, nous réduisons considérablement le nombre d'états générés et conservés, tout en assurant l'optimalité de la fonction récursive (28). Dans le contexte de génération de colonnes où ce problème de plus court chemin avec contraintes supplémentaires est résolu des centaines de fois, nous avons besoin d'un algorithme rapide et demandant peu d'espace mémoire. Dans la section suivante nous allons encore modifier la récursion (28) pour rendre l'algorithme générateur plus rapide mais ceci au dépend de l'optimalité de la solution du sous-problème. Notons que nous pouvons atteindre l'optimalité finale du problème maître sans toujours résoudre les sous-problèmes de façon optimale, il suffit d'obtenir une colonne de coût relatif négatif à chaque itération.

3.3 Approches heuristiques

L'efficacité de la technique de programmation dynamique dépend essentiellement du nombre d'états générés à chaque étape. Limitons donc les états de façon astucieuse.

3.3.1 Relaxation de la contrainte d'élimination de cycles

Puisque l'application de la contrainte d'élimination de cycle rend la résolution plus difficile, nous la relaxons partiellement en permettant des routes avec certains cycles.

Il s'agit de trouver une application surjective qui réduira la cardinalité du domaine Σ (voir définition page 24) des états possibles à une image Σ' où $|\Sigma'| <<< |\Sigma|$. De cette façon le nombre d'états à chaque étape restera plus petit.

Soit $\bar{g}$ l'application surjective qui associe à un ensemble S, un de ses sous-ensembles : $\bar{g}(S) \subseteq S$.

La récursion (28) devient:

$$(31)\quad f_k(\bar{g}(S),j,t_j) = \min_{i\in h'_j(S)} \{ f_{k-1}(\bar{g}(S-\{j\}),i,t_i)+c_{ij}(C(\bar{g}(S))) \mid t_j \geq t_i + t_{ij} + M_i,\ a_j \leq t_j \leq b_j,\ C(\bar{g}(S)) \leq C^q,\ Pr(j) \subseteq \bar{g}(S) \}$$

où $\bar{g}(S) = \{i^+ \mid i^+ \in S \text{ et } i^- \notin S\}$

et $h'_j(S) = \{i \in \bar{g}(S) \cup P^- \mid (i,j) \in I\}$.

$\bar{g}(S)$ est l'ensemble des noeuds d'origine visités dont les destinations correspondantes ne sont pas encore visitées et $h'_j(S)$ est l'ensemble des noeuds prédécesseurs possible de j étant donné $\bar{g}(S)$ (notons que $h_j(S) \subseteq h'_j(S)$ puisque $S \subseteq \bar{g}(S) \cup P^-$).

Au lieu de conserver l'ensemble S de tous les noeuds visités, nous ne conserverons qu'un sous-ensemble de noeuds visités. On peut facilement se convaincre que l'ensemble $\bar{S}(S)$ est suffisant pour permettre de vérifier les contraintes de compatibilité, d'intervalle de temps, de capacité et de précédence. Mais par contre cette relaxation ne permet plus d'éviter de produire des routes sans cycle: en effet $\bar{S}(S)$ ne contient aucune information sur les demandes dont leur origine et leur destination ont été visitées.

Par contre, pour que de tels cycles se produisent, il faut que le réseau du sous-problème contiennent des cycles de coûts négatifs (après l'ajout des variables duales). Et une route contenant un tel cycle sera toujours d'un coût et d'une durée finis à cause des intervalles de temps bornés. Remarquons que la récursion (31) permet de produire des routes avec des cycles d'une longueur de 2 noeuds. Mais en rajoutant la contrainte $Pr(i) \neq \{j\}$ à h'_j, nous évitons les cycles de longueur 2 et les cycles de longueur 3 sont parcourus au plus une fois. Avec cette contrainte supplémentaire les cycles deviennent beaucoup plus rares.

Si une telle route avec cycle est produite, la colonne correspondante est tout de même ajoutée au problème maître restreint avec des coefficients supérieurs à 1 pour les demandes visitées plus d'une fois. La variable associée peut entrer dans la base à un niveau fractionnaire mais redevient nulle par la suite par une itération du simplexe ou par branchement car elle ne peut prendre la valeur 1 dans la solution optimale entière du problème maître.

En somme, cette relaxation réduit de beaucoup le nombre d'états conservés et générés et est d'une grande importance.

Remarquons de plus que la solution obtenue avec cette nouvelle récursion fournit une borne inférieure à celle obtenue avec la récursion (28) et peut être utilisée en permanence sans affecter l'optimalité en nombre entier du problème maître.

3.3.2 Restriction sur h'_j

Cette restriction consiste à réduire l'ensemble $h'_j(S)$ de la récursion (31). Elle élimine certains arcs peu probables. Bien entendu cela permet à l'algorithme d'être beaucoup plus efficace mais nous perdons la borne inférieure que nous avions et qui servait de critère d'arrêt pour l'optimalité du problème maître. C'est pourquoi cette restriction est temporaire et sert uniquement à générer les premières colonnes du problème maître restreint (voir section 4.3). De plus, si cette restriction est astucieuse, elle permettra de générer la majorité des colonnes nécessaires au problème maître restreint pour l'optimalité du problème maître en peu de temps.

La récursion prend la forme suivante:

$$(32)\quad f_k(\underline{S}(S),j,t_j) = \underset{i\in h''_j(S)}{\text{MIN}} \{ f_{k-1}(\underline{S}(S-\{j\}),i,t_i)+c_{ij}(C(\underline{S}(S))) \mid t_j \geq t_i + t_{ij} + M_i,\ a_j \leq t_j \leq b_j,\ C(\underline{S}(S)) \leq C^q,\ Pr(j) \subseteq \underline{S}(S) \}$$

où $h''_j(S) = \{i\in\underline{S}(S) \cup P^- \mid (i,j)\in I' \subseteq I\} \subseteq h'_j(S)$.

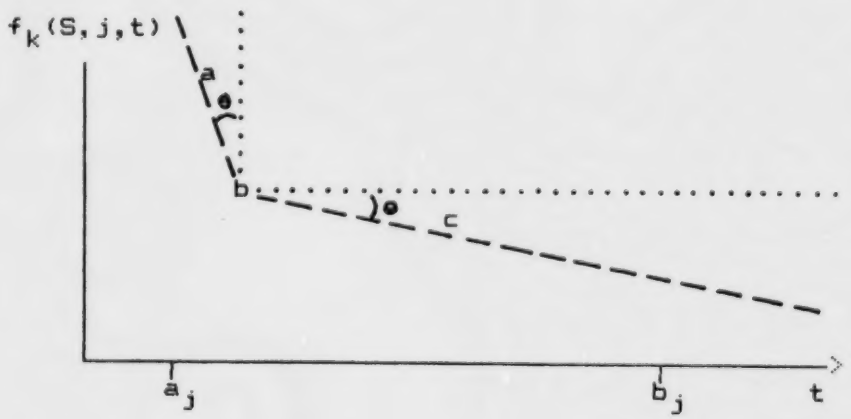
L'ensemble $h''_j(S)$ est constitué des prédécesseurs de j les plus probables. Le nouvel ensemble I' des arcs du réseau sera expliqué dans la section 4.3.

3.3.3 Restriction du nombre d'états par dominance augmentée

Nous avons montré, dans la section 3.2, que la dominance est représentée par le quadrant supérieur droit. Cette dominance nous permet de ne conserver que les états important pour assurer l'optimalité du sous-problème. Pour les problèmes où le nombre d'états conservés est trop grand, il devient important de pouvoir considérer seulement les états les plus intéressants. La réduction de l'ensemble des états permettra alors de réduire la taille du problème mais encore ici, nous pourrons perdre l'optimalité de la solution du sous-problème.

Cette réduction se fait en élargissant la surface de dominance des états. Cette nouvelle dominance augmentée est présenté par la figure 3.2.

Figure 3.2 : Graphique sur la dominance augmentée



La surface du quadrant de dominance est élargie de part et d'autre par un angle θ. Nous pouvons voir que les états représentés par les points "a" et "c" sont dominés par celui représenté par le point "b". La dominance augmenté d'un angle θ

se formule alors comme suit: l'état (S,j,t) domine l'état (S,j,t') si (33) est satisfait.

$$(33)\quad \begin{cases} f_k(S,j,t') - f_k(S,j,t) \geq tg(\theta) \cdot (t - t') \\ \text{et} \quad t' - t \geq tg(\theta) \cdot (f_k(S,j,t) - f_k(S,j,t')) \end{cases}$$

Cette technique est employée seulement lorsque le problème traité génère trop d'états.

3.4 Génération de plusieurs colonnes à la fois

Dans le cadre de la technique de génération de colonnes, pour un même type de véhicules, le sous-problème peut être appelé plusieurs fois pour générer des routes mutuellement exclusives (voir section 4.2). D'une fois à l'autre nous enlevons du sous-problème les demandes de la route générée. Nous nous assurons ainsi d'avoir des routes mutuellement exclusives à moindre coût parmi les noeuds non utilisés par les routes précédentes.

Au lieu de résoudre le sous-problème plusieurs fois, nous voudrions à partir de la relation (29) , non pas trouver la meilleure route à moindre coût mais les meilleures routes mutuellement exclusives à moindre coûts.

Formellement nous voulons trouver :

$$(29)\quad f_k^*(S^*,g^-,t^*) = \underset{\substack{(S,j,t_j)\in\Sigma \\ k=2,4,6,\ldots}}{\text{MIN}} \{ f_k(S,j,t_j) + c_{jg^-}(0) \mid a_{g^-} \leq t^* \leq b_{g^-},\ t^* \geq t_j + t_{jg^-} + M_j,\ C(S)=0 \}$$

$$(34)\quad f_{k'}(S',g^-,t') = \underset{\substack{(S,j,t_j)\in\Sigma \\ k=2,4,6,\ldots}}{\text{MIN}} \{ f_k(S,j,t_j) + c_{jg^-}(0) \mid a_{g^-} \leq t' \leq b_{g^-},\ t' \geq t_j + t_{jg^-} + M_j,\ C(S)=0,\ S \cap S^* = \emptyset \}$$

$$(35)\quad f_k''(S'',g^-,t'') = \underset{\substack{(S,j,t_j)\in\Sigma \\ k=2,4,6,\ldots}}{\text{MIN}} \{f_k(S,j,t_j)+c_{jg^-}(0) \;:\; a_{g^-} \le t'' \le b_{g^-},\ t'' \ge t_j + t_{jg^-} + M_j,\ C(S)=0,\ S \cap S^* = \emptyset,\ S \cap S' = \emptyset\}$$

.
.
.

Clairement (S^*,g^-,t^*) est l'état optimal du problème, (S',g^-,t') est optimal pour le problème où les demandes déservies par la première route ont été enlevées et (S'',g,t'') est optimal pour le problème où les demandes déservies par les deux premières routes ont été enlevées, etc. Si (34) et (35) pouvaient être calculées facilement, nous sauverions 2 appels au sous-problème.

Mais (34) et (35) ne peuvent être calculées simultanément avec (28), puisque (34) et (35) supposent que l'on connaît l'état optimal (S^*,g^-,t^*). Il faudrait donc construire l'ensemble des routes réalisables de la recursion (29): $\Sigma' = \{(S,j,t) \;:\; (S,j,t)\in\Sigma \text{ et } C(S)=0\}$ et ensuite nous pourrions appliquer (7) et (8) sur Σ'. Mais encore là Σ' est de trop grande taille pour conserver cet ensemble. Puisque l'optimalité est trop difficile à atteindre pour les équations (34), (35), ... contentons nous de trouver si possible de bonnes solutions pour elles. Construisons donc un sous-ensemble de Σ' en effectuant une partition des routes de Σ' selon la dernière destination visitée avant le retour au dépôt et en ne conservant que les états correspondants à la route de coût minimum dans chaque partie. Remarquons que ce sous-ensemble ne contient que n éléments (pour n demandes). Ce sous-ensemble est défini par

$$\{(S',j,t') \;:\; j\in P^- \text{ et } f_k,(S',j,t') = \underset{\substack{(S,j,t)\in\Sigma' \\ k=2,4,6,\ldots}}{\text{MIN}} [f_k(S,j,t)]\}$$

Ce sous-ensemble est petit et peut être construit simultanément avec la récursion (28). Alors nous appliquons les récursions (34), (35) etc. à ce sous-ensemble.

Avec les relaxations et restrictions introduites précédemment la même technique peut être employée dans le cas des récursions (31) et (32) mais les formules (34), (35), ... devront être modifiées puisque pour ces deux récursions l'ensemble S est relaxé par la fonction $\bar{g}(\cdot)$ et que si C(S)=0 alors S=∅. Nous remplacerons alors $S \cap S^*$ par $\bar{g}^{-1}(S,j,t_j) \cap \bar{g}^{-1}(S^*, g^-, t^*)$ où $\bar{g}^{-1}$ est une fonction qui retourne l'ensemble des noeuds visités de la meilleure route de l'état fourni. Un remplacement équivalent est fait pour $S \cap S'$.

Comme nous pourrons le voir dans le chapitre 7, cette technique fournit de bons résultats.

CHAPITRE 4

RESOLUTION DU PROBLEME MAITRE

4. RESOLUTION DU PROBLEME MAITRE

Nous avons vu au chapitre 2 les étapes de base de l'algorithme de la génération de colonnes pour résoudre le problème maître. Ce problème est en soit très difficile à résoudre et nous a forcé à développer une approche permettant d'accélérer la convergence de l'algorithme.

L'approche développée se base sur l'idée suivante: si un bon problème maître restreint est construit à l'étape 0 (voir section 2.2.1) alors très peu d'itérations seront nécessaires afin d'obtenir une solution optimale du problème maître. Cette approche de construction d'un bon problème maître restreint est discutée dans la section 4.3.

Nous avons également utilisé différentes stratégies afin d'accélérer la rapidité de l'algorithme. Ces stratégies sont utilisées dans les étapes 1 et 2 et sont discutées dans les prochaines sections.

4.1 Etape 1 : résolution du problème maître restreint

Le problème maître restreint est résolu par un algorithme du simplexe qui permet la réoptimisation et l'obtention des variables duales nécessaires aux sous-problèmes.

Les auteurs de [10] ont proposé plusieurs stratégies permettant une réduction du temps d'exécution de cette étape. La première stratégie consiste à remplacer dans l'équation (15) la condition d'égalité (=) par une inégalité (≥) (i.e. résoudre un problème de recouvrement). Ceci est possible puisque chaque sous-route d'une route réalisable est réalisable à moindre coût et qu'à l'optimalité l'équation (15) tient nécessairement avec égalité. La résolution du problème de recouvrement permet entre autres une meilleure stabilité numérique. Une deuxième statégie

proposée consiste à perturber le terme de droite des contraintes du problème maître restreint pour réduire la dégénérescence laquelle est en autre très élevée dans les problèmes de routage. La dernière stratégie consiste à éliminer les variables dont le coût relatif est très positif. Cela permet de conserver un problème maître restreint de taille raisonable.

Pour nos applications, seule la première stratégie fut utilisée, les deux autres ne donnant aucun résultat satisfaisant. Notre problème maître restreint étant toujours petit la perturbation n'apporte aucun avantage. De plus puisque chaque colonne est difficile à générer et que le nombre de colonnes est relativement petit, l'élimination augmente le temps d'exécution puisque certaines colonnes éliminées demandent à être regénérées.

4.2 Etape 2 : résolution des sous-problèmes

La principale tâche de cette étape est de générer la prochaine colonne (non disponible) à entrer dans la base du problème maître restreint afin de diminuer la fonction économique le plus possible. Cette colonne est celle du plus petit coût relatif. La recherche de cette colonne consiste, comme nous l'avons vu, à résoudre $|V|$ sous-problèmes. Mais les premiers essais nous ont montré que la résolution des sous-problèmes consomme beaucoup de temps d'ordinateur.

Une stratégie permettant d'accélérer la convergence consiste à fournir, à chaque étape, plusieurs colonnes susceptibles de diminuer l'objectif (colonnes de coût relatif négatif) du problème maître restreint (Desrosiers, Soumis et Desrochers [9]). Cette stratégie exploite l'idée qu'il est bon d'inclure dans l'ensemble des routes du problème maître restreint, toutes routes qui ont une probabilité d'être

optimales et même des routes non optimales qui peuvent permettre au cours des itérations suivantes de réduire l'objectif et d'améliorer les variables duales.

Cette stratégie se voit efficace lorsque les colonnes fournies sont disjointes puisqu'en ne partageant pas les mêmes ressources ils peuvent coexister plus facilement dans une base et réduire davantage l'objectif. Ceci est réalisé en enlevant du réseau, après avoir généré une colonne, les noeuds utilisés par celle-ci. La prochaine colonne générée est alors nécessairement disjointe des précédentes et plus facilement générée étant donné que le réseau est réduit.

De multiples colonnes sont obtenues soit par l'algorithme des sous-problèmes (puisque nous avons montré au chapitre 3 que l'algorithme peut générer plusieurs colonnes disjointes simultanément), soit en résolvant plusieurs sous-problèmes ou soit en résolvant plusieurs fois le même sous-problème (surtout pour le cas $|V| = 1$). Notons que pour le cas $|V| > 1$, nous ne sommes plus assurés, en générant les colonnes de cette façon, que la colonne de plus petit coût relatif soit parmi celles générées. Ceci est de moindre importance puisque la convergence à l'optimalité est assurée même si l'étape 2 ne fournit pas la colonne de plus petit coût relatif. En effet, il suffit qu'une colonne de coût relatif négatif soit générée à chaque itération pour assurer la convergence. Dans nos applications, nous nous sommes limités à fournir au plus 3 à 4 colonnes susceptibles de diminuer l'objectif.

Une autre façon de diminuer le temps consommé est d'exploiter les remarques suivantes:

(1) Si la variable duale σ_i $i \in P^+$ (du chapitre 2) est près de zéro alors la demande i associée ne fera pas partie d'un plus court chemin. En effet: si le plus court chemin de

coût β, d'un sous-problème, inclus la demande i qui occasionne un déplacement suplémentaire τ (non nul) alors le même chemin en excluant les noeuds de la demande i a un coût de $\beta-\tau+\sigma_i$ où $\tau > 0$, $\sigma_i \approx 0 \Longrightarrow -\tau+\sigma_i < 0$ ce qui contredit le fait que le chemin de coût β est le plus court.

(2) (extention heuristique de la remarque (1))

Si $\sigma_i <$ B(i) = "une fonction du coût de déplacer la demande i" alors la demande i ne devrait probablement pas se trouver sur le plus court chemin.

La remarque (1) exprime que servir la demande i occasionne un désavantage τ et aucun avantage (σ_i). De ce fait on peut réduire le réseau selon les variables duales obtenues du problème maître restreint à chaque itération majeure en éliminant les noeuds et les arcs associés aux demandes éliminées par l'application de la remarque (1).

La remarque (2) exprime que si la variable duale σ_i n'absorbe pas en partie le coût de déplacer la demande i (incluant le coût fixe d'utilisation du véhicule et du déplacement de la demande i) alors, celle-ci ne fera probablement pas partie d'un plus court chemin. Lorsque les variables duales sont très peu ajustées, il est facile et profitable de réduire le réseau en éliminant les noeuds des demandes associées aux variables duales respectant la remarque (2).

L'exploitation de la deuxième remarque n'est utilisée que temporairement pendant les premières itérations de l'algorithme de génération de colonnes tandis que la première est utilisée en permanence sauf qu'elle n'est pas appliquée sur les demandes qui sont fixées dans l'arbre de SEP.

4.3 Construction d'un bon problème maître restreint

Une façon d'obtenir un problème maître restreint de départ est de le construire avec des colonnes artificielles. Ensuite en appliquant l'algorithme de génération de colonnes sur l'ensemble I des arcs du réseau nous résolvons alors le problème maître avec généralement beaucoup de longues itérations.

Si au lieu d'appliquer l'algorithme sur les arcs de I on l'appliquait sur un petit sous-ensemble I' alors le problème maître restreint final obtenu en serait un de départ pour résoudre le problème maître sur l'ensemble des arcs de I. Un bon choix d'un sous-ensemble I' permettra de construire rapidement un problème maître restreint qui fournira une solution optimale près de celle du problème maître.

Pour construire le sous-ensemble I', nous verrons au chapitre sur l'implantation que nous avons associé à chaque arc (i,j) un poids P(i,j) qui donne une appréciation sur la qualité (en coût) du déplacement occasionné par celui-ci, que les arcs ont été groupés par prédécesseur et qu'ils ont été triés selon leur poids dans chacun des groupes, fournissant ainsi pour chaque arc son rang centile R(i,j) dans son groupe. Ces deux mesures permettent de réduire l'ensemble I. Le sous-ensemble I', obtenu par l'utilisation de ces mesures dans l'équation (36), inclut les arcs ayant un poids inférieur au poids moyen de tous les arcs ($\bar{P}$) et un rang centile inférieur à 55. Cet ensemble inclut également des arcs supplémentaires, de façon à ce que pour chaque noeud nous ayons un nombre minimum d'arcs sortant. Pour 11 problèmes, la cardinalité du sous-ensemble I' obtenue est d'environ la moitié de celle de l'ensemble I et ce sous-ensemble I' permet d'obtenir l'optimalité du problème maître dans 75 % des cas. Dans les autres cas la solution était très proche de l'optimalité.

(36) $I' = \{(i,j) \in I \mid P(i,j) \leq \bar{P},\ R(i,j) \leq 55\ \%\}$

Il est possible de faire mieux encore en ne construisant pas le problème maître restreint en une seule étape mais bien en plusieurs étapes. Il s'agit de partir d'un petit réseau qu'on grossit d'étape en étape jusqu'à l'obtention du réseau total. Au chapitre des résultats nous discuterons du comportement de l'algorithme d'étape en étape.

CHAPITRE 5

RESOLUTION DU PROBLEME MAITRE EN NOMBRES ENTIERS

5. RESOLUTION DU PROBLEME MAITRE EN NOMBRES ENTIERS

Puisque nous avons relaxé la contrainte d'intégralité (17) de la nouvelle formulation présentée au chapitre 2, nous ne sommes plus assurés que la résolution du problème maître fournisse une solution optimale entière. De plus, les variables présentes dans le problème maître restreint final peuvent être insuffisantes pour l'obtention de la solution optimale entière d'où la possibilité de générer de nouvelles colonnes.

On favorise l'apparition de solutions entières en utilisant une coupe exigeant l'utilisation d'un nombre entier de véhicules. Quand après celle-ci nous n'obtenons pas de solution entière, on utilise alors un algorithme de séparation et d'évaluation progressive (SEP) où à chaque noeud de l'arbre d'énumération un problème maître restreint est réoptimisé et des colonnes sont générées au besoin.

5.1 Coupe sur le nombre de véhicules

Comme nous en avons discuté précédemment, un coût fixe d'utilisation du véhicule est ajouté au coût de déplacement de chaque route. De grands coûts fixes nous assurent que le nombre de véhicules est minimisé en priorité. De cette façon, si la solution optimale du problème maître utilise un nombre fractionnaire τ de véhicules, nous sommes alors assurés que la solution optimale entière utilise au moins $\lceil\tau\rceil$ véhicules où $\lceil\ \rceil$ signifie le plus petit entier supérieur ou égal. Nous rajoutons donc au problème maître restreint une nouvelle contrainte demandant d'utiliser au moins $\lceil\tau\rceil$ véhicules. La nouvelle contrainte est la suivante:

$$\sum_{g\in V}\ \sum_{r\in\Omega_g} x_r^g \geq \lceil\tau\rceil \qquad (37)$$

Lorsque nous avons des véhicules dont certains avec coût fixe et d'autres sans coût fixe alors la coupe s'applique uniquement aux véhicules avec coût fixe.

Après l'addition de cette coupe le problème maître restreint est réoptimisé et de nouvelles colonnes sont générées si nécessaire. La fonction objective des sous-problèmes est modifiée puisqu'une nouvelle variable duale α associée à la contrainte (37) modifie les coûts relatifs. L'objectif d'un sous-problème $g \in V$ devient:

$$\text{MIN } Z = \sum_{(i,j)\in I} [\, c^g_{ij}(y^g_i) - \sigma_i \,] \cdot x^g_{ij} - \pi_g - \alpha$$

Il arrive fréquemment après avoir effectué cette coupe sur le nombre de véhicules que la solution optimale du simplexe est une solution optimale entière. Lorsque ce n'est pas le cas, on utilise une procédure de SEP.

5.2 Séparation et évaluation progressive (SEP)

Une technique de SEP se caractérise principalement par le choix d'une variable à fixer, une règle de séparation, une stratégie de branchement et l'obtention de bornes inférieures. Décrivons donc ces caractérisques pour notre algorithme de SEP.

Choix de la variable: Parmi les variables fractionnaires, nous choisissons celle ayant un bon compromis entre son coût et son influence sur les autres variables dans la base si celle-ci était fixée (i.e. on cherche une variable avec un petit coût et qui partage le plus d'éléments avec les autres variables non nulles de la base courante). Par ce critère nous influençons la base courante et favorisons l'apparition d'une nouvelle base avec probablement plus de variables entières et à un coût à peu près semblable.

Règle de séparation: Le branchement ne s'effectue pas en fixant les variables X_r^g à 0 ou à 1 mais bien en fixant les k variables x_{ij}^g associées aux arcs qui composent la route r. Les 2^k façons de fixer ces variables sont regroupées en k+1 branches comprenant respectivement 1, 1, 2, 4, ..., 2^{k-1} façons. La première branche fixe les k arcs à 1. La seconde fixe les k-1 premiers arcs à 1 et le dernier à 0. Ainsi de suite jusqu'à la $k+1^{ième}$ qui fixe le premier à 0.

Soit les variables suivantes (définissant la route r) associées à la variable $0 < X_r^g < 1$:

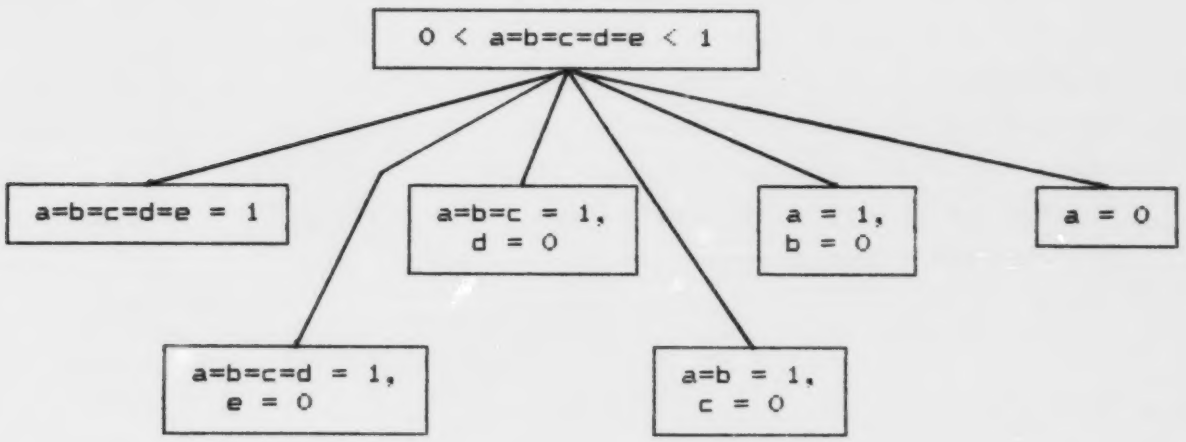
$$x_{g\,i}^{g+} = x_{i\,j}^{g+} = x_{j\,j}^{g+} = x_{j\,i}^{g-} = x_{i\,g}^{g-} = X_r^g$$

que nous noterons, pour fin de simplification:

$$a = b = c = d = e$$

La figure 5.1 présente la façon dont ce fait la séparation.

Figure 5.1 : L'arbre d'énumération implicite



Cette partition d'un noeud correspond à la méthode introduite par Bellmore et Malone [2] et améliorée par Carpeneto et Toth [5] pour le voyageur de commerce. Ce type de partition

permet, dans notre méthodologie de génération de colonnes, de conserver dans chaque branche un problème maître et des sous-problèmes de même type que ceux du départ, ce qui ne serait pas possible si on partitionnait en fixant les variables x_r^q à 0 ou 1.

Stratégie de branchement: Les noeuds de l'arbre d'énumération sont traités de gauche à droite et de bas en haut. Cette descente sur la gauche réduit la taille du problème et trouve rapidement une solution entière. Dans chaque branche on modifie le réseau en éliminant les arcs fixés à 0 et les arcs rendus impossibles par ceux fixés à 1. Les colonnes du problème maître restreint utilisant des arcs éliminés sont enlevées. Le problème maître restreint est alors réoptimisé et de nouvelles colonnes sont générées au besoin à partir du réseau modifié.

L'obtention de bornes inférieures: La borne inférieure à chaque noeud est fournie par la résolution du problème linéaire en nombres réels. Dans les cas où l'on veut accélérer la technique de SEP au dépend de l'optimalité, on peut éliminer les noeuds pour lesquels leur borne inférieure ne promet pas d'améliorer le déplacement de la meilleure solution entière trouvée d'un certain pourcentage. Dans nos applications nous avons utilisé un pourcentage égal à 0.1 % fois le nombre de noeuds déjà explorés, exprimant ainsi que plus on explore de noeuds dans l'arbre, moins on est exigeant.

CHAPITRE 6

IMPLANTATION

6. IMPLANTATION

Bien que l'efficacité de la résolution d'un problème dépende essentiellement de la méthodologie utilisée, il n'en demeure pas moins que son implantation peut être déterminante. Une mauvaise implantation peut faire en sorte qu'une bonne méthode de résolution soit inefficace. C'est pourquoi une attention particulière y est accordée.

Les principaux facteurs influençant notre méthodologie sont les sous-problèmes et le réseau. Les sous-problèmes influencent par leur temps de résolution et l'espace mémoire utilisé, tandis que le réseau influence, de par sa construction, le temps de résolution des sous-problèmes.

La résolution du problème maître restreint n'est pas en soit un gros facteur d'influence. Sa taille est petite dans les aplications que nous avons traitées et peu de temps est nécessaire à sa résolution. L'algorithme du simplexe utilisé doit toutefois permettre la réoptimisation.

Nous allons présenter l'algorithme et la structure de données principale, développés pour le sous-problème et la construction du réseau utilisé.

6-1 L'algorithme de programmation dynamique

L'algorithme de programmation dynamique a son importance pour l'efficacité de la résolution d'un sous-problème; nous en donnons ses principales étapes.

Soit g^+ et g^- les noeuds de départ et de retour du véhicule au dépôt et I' l'ensemble des arcs permis du réseau.

Les étapes de l'algorithme sont les suivantes:

étape 0 : Refermer les intervalles de temps de la façon suivante:

- $b_i^- = \text{MIN}\ (b_i^-,\ b_g^- - t_{i-g}^- - M_i^-)$
- $b_i^+ = \text{MIN}\ (b_i^+,\ b_i^- - t_{i+i}^- - M_i^+)$

Cette façon de refermer les fenêtres nous assure que si une demande i est réalisable dans le temps, alors de celle-ci nous pouvons nous rendre au dépôt de retour du véhicule avant l'heure b_g^-.

étape 1 : Initialisation:

- $\Sigma = \emptyset$
- Poser pour $i \in P^+$:
  $f_1(\{i\},i,t) = c_{g+i}(0)$ où $t = \text{MAX}\ (a_i,\ a_g^+ + t_{g+i})$
  et faire $\Sigma = \Sigma \cup \{(\{i\},i,t)\}$
- $\Sigma' = \emptyset$
- $k = 1,\ Z = \infty,\ kmax = 10.$

Par la suite, Σ représentera l'ensemble des états conservés de l'étape précédente, Σ' celui de l'étape en cours, k le nombre d'itérations effectuées, kmax le nombre maximum d'itérations permis et Z la meilleure solution trouvée.

étape 2 : Construction des états définissant les routes de longueur k noeuds:

2.1 : Si $k \geq kmax$ alors procéder à l'étape 4.
2.2 : Si $\Sigma = \emptyset$ alors procéder à l'étape 3.
2.3 : Soit $(S,i,t) \in \Sigma$ ...(note 1)

2.4 : Construction des états réalisables émergeant de (S,i,t):

Pour tout j tel que $(i,j) \in I'$ faire

- Si $j \in S$ ou $Pr(j) \notin S$ ou $Pr(i) = \{j\}$ ou $C(S \cup \{j\}) > C^9$ alors procéder à l'étape 2.4 avec un nouveau j ...(note 2)

(note 1)

- $t' = MAX\ (a_j,\ t + t_{ij} + M_i)$
- Si $t' > b_j$ alors procéder à l'étape 2.4 avec un nouveau j ...(note 2)
- $S' = \xi(S \cup \{i\})$
- Si (S',j,t') est non post-réalisable alors procéder à l'étape 2.4 avec un nouveau j ..(note 5)
- $f_{k+1}(S',j,t') = f_k(S,i,t) + c_{ij}(C(S \cup \{j\}))$..(note 3)
- $\Sigma' = \pi(\Sigma' \cup \{(S',j,t')\})$ ...(note 4)
- Si $C(S') = 0$ et $f_{k+1}(S',j,t') + c_{jg}-(0) < Z$ alors $Z = f_{k+1}(S',j,t')$
- Procéder à l'étape 2.4 avec un nouveau j

2.5 : $\Sigma = \Sigma - \{(S,i,t)\}$ et retourner à l'étape 2.2

étape 3 : Réinitialiser la prochaine étape 2:

- Si $\Sigma' = \emptyset$ alors procéder à l'étape 4
- $\Sigma = \Sigma'$
- $\Sigma' = \emptyset$
- $k = k + 1$
- Procéder à l'étape 2.

étape 4 : Terminer l'algorithme:

- Si $Z > \infty$ alors le problème n'a pas de solution réalisable et nous terminons.
- sinon, Z est la meilleure solution trouvée et nous retournons la route qui a permis de construire Z.
- Fin de l'algorithme.

notes:

(1) Les parties encadrées sont en réalité plus générales dans notre implantation et elles permettent d'accélérer l'algorithme. Nous ne traitons pas seulement un élément de Σ à la fois mais un sous-ensemble $\Sigma_{S,i} \subseteq \Sigma$ simultanément où $\Sigma_{S,i} = \{(S,i,t') \in \Sigma\}$ i.e. nous traitons simultanément tous les les états ne se distinguant que par leur temps d'arrivée au noeud terminal i (ou tout simplement les points de discontinuité) (Référer à la section 3.2 et Desrosiers, Soumis et Pelletier [11]).

(2) Ces conditions représentent la vérification de la réalisabilité des contraintes du problème. (Référer à la section 3.1)

(3) $f_k(S,i,t)$ représente le coût minimal de l'état (S,i,t) mais $f_{k+1}(S',j,t')$ ne le représente pas pour l'état (S',j,t') tant que l'étape 2 n'est pas terminée.

(4) π consiste à ne conserver que les états qui définissent les points de discontinuité de la fonction f (Référer à la section 3.2) i.e. $\pi(\Sigma) = \Sigma'$ implique que si $(S,i,t') \in \Sigma$ alors $(S,i,t') \in \Sigma'$ s'il n'existe pas $(S,i,t) \in \Sigma$ tel que $f_1(S,i,t) \leq f_1(S,i,t')$ et $t \leq t'$ avec au moins une inégalité stricte.

(5) Le test de non post-réalisabilité est le suivant: Pour $k,l \in T'$ tel que $k \neq l$, $|T'| \leq 2$ et $T' \subseteq T = \{i^- \mid i^+ \in S'\}$ nous avons que:

Si $MAX(t' + t_{jk} + M_j, a_k) + t_{kl} + M_k > b_l$ et

si $MAX(t' + t_{jl} + M_j, a_l) + t_{lk} + M_l > b_k$

alors l'état (S',j,t') est non post-réalisable.

En d'autres mots, lorsque ces conditions sont vérifiées, (S',j,t') est non post-réalisable puisqu'il n'existe pas de façon de terminer la route en ayant tout livré la

marchandise dans le véhicule dans les intervalles de temps permis.

Remarques:

(1) L'étape 2.4 est la plus cruciale, c'est la partie la plus interne de l'algorithme. Les opérations majeures sont le test de la non post-réalisabilité et le calcul de $\pi(\cdot)$. Le nombre de ces opérations est de l'ordre de $|\Sigma| \cdot n$ pour chaque étape 2.

(2) L'algorithme ne conserve que Σ et Σ' simultanément en mémoire. Et remarquons que nous diminuons l'ensemble Σ à l'étape 2.5 de façon à ce que $|\Sigma| + |\Sigma'|$ reste petit.

6.2 Structure de données principale

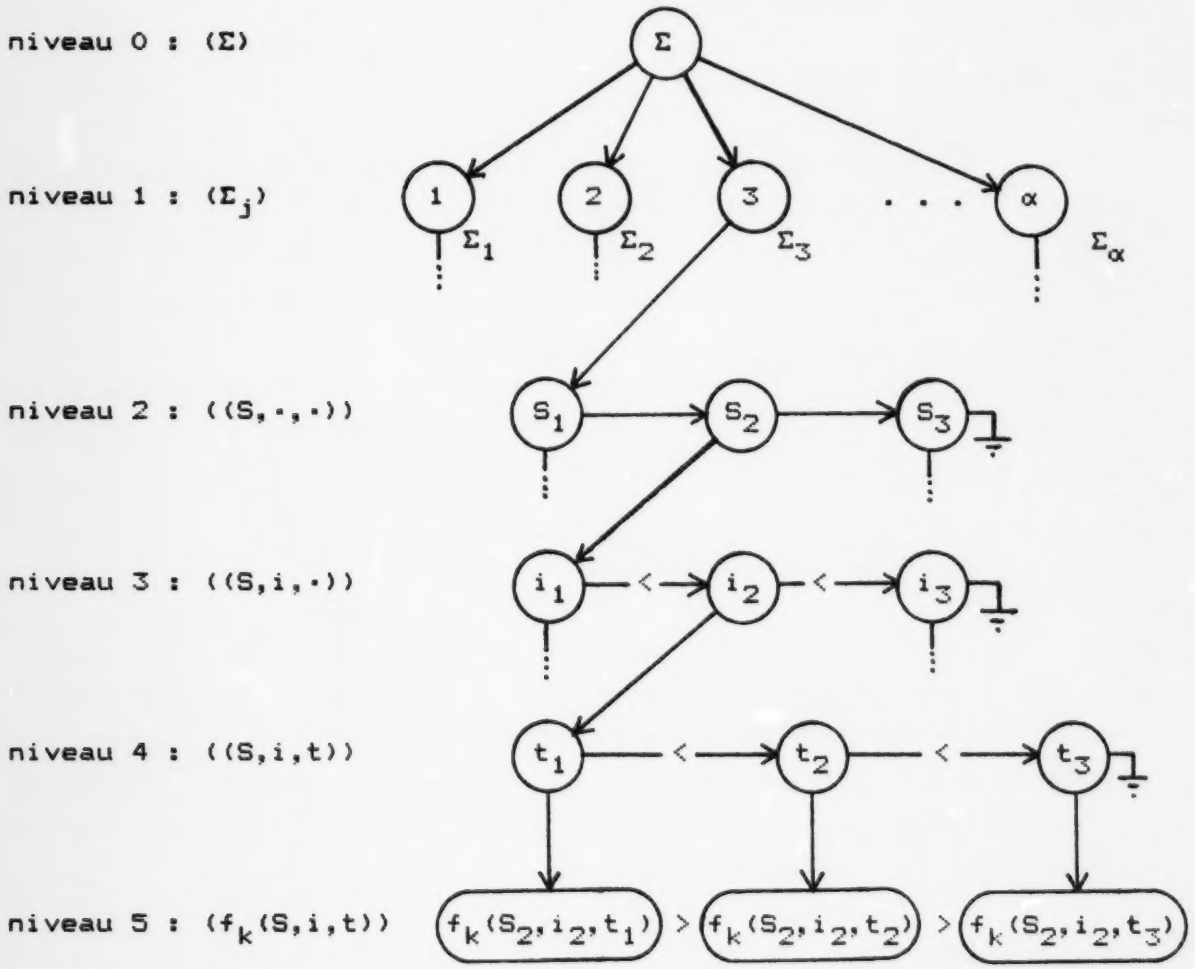
Les principales difficultés que nous retrouvons dans l'algorithme présenté ci-haut sont la gestion et la définition de l'ensemble Σ. Nous avons donc développé une structure de données pour cet ensemble, permettant: l'utilisation de peu d'espace mémoire, un calcul rapide de $\pi(\Sigma)$ et l'addition et l'élimination facile d'un élément.

La structure développée est celle d'un arbre et elle se définit comme suit:

Soit $\Sigma = \{(S,i,t) \mid (S,i,t)$ est un état réalisable$\}$ et $\Sigma_i = \{(S,i,t) \mid (S,i,t) \in \Sigma$ et le modulo α de S est i-1$\}$. Nous avons alors $\Sigma_i \cap \Sigma_j = \emptyset$ et $\bigcup_i \Sigma_i = \Sigma$ i.e. une partition des états réalisables.

La structure de l'arbre est illustrée à la figure 6.1.

Figure 6.1 : Structure de l'arbre des états



La recherche d'un état (S,i,t) se fait donc de la façon suivante: à partir du niveau 0 nous déterminons à quelle partie appartient S, nous passons alors au niveau 1 qui lui nous envoie directement au niveau 2. Par une recherche séquentielle nous parcourons la liste enchaînée des ensembles S' distincts jusqu'à ce que S soit trouvé (s'il existe). Le noeud S trouvé nous envoie alors au niveau 3. Par une recherche séquentielle partielle (puisque les éléments sont triés) nous cherchons i. Le noeud i trouvé (s'il existe) nous envoie alors au niveau 4 d'où

encore une fois une recherche séquentielle partielle est effectuée pour trouver t.

A chaque extrémité de cet arbre (niveau 5), nous garderons, entre autre, une information supplémentaire: soit l'ordre de visite des noeuds de la route qui a engendré le coût de l'état. Cette information est nécessaire pour retracer la route de chaque état.

Pour un calcul rapide de $\pi(\Sigma)$, nous avons regroupé sous un même toit tous les états ayant le même ensemble S. De la même façon, nous avons regroupé ceux ayant le même ensemble S et le même noeud terminal. De cette manière, π est facilement calculé et l'espace mémoire utilisé est minimum. Remarquons également que l'addition et l'élimination d'un élément se fait assez facilement par une gestion efficace des pointeurs.

L'étape 2 de l'algorithme présenté dans la section 6.1 dépend essentiellement de $|\Sigma|$ (voir remarque (1) dans la section 6.1). Evaluons donc l'ordre de grandeur de la cardinalité maximum de cet ensemble Σ dans le cas particulier où les quantités de marchandise (d_i) à transporter sont parmi l'ensemble $\{C^*, 1/2 \cdot C^*, 1/3 \cdot C^*, 1/4 \cdot C^*\}$.

Définissons:

D_r le nombre de demande avec charge $\leq (1/r) \cdot C^*$

et N_i le nombre de noeuds du niveau i de l'arbre des états (figure 6.1).

Alors $N_0 = 1$,

$N_1 = \alpha$,

$$N_2 = \text{MAX}\left[\binom{D_3}{3} + \binom{D_1}{1}, \binom{D_4}{4} + \binom{D_2}{2} + 1\right],$$

$N_3 = N_2 \cdot n$,

$N_5 = N_4 = N_3 \cdot T$ où

n est le nombre de demandes, T le nombre maximum de points de discontinuité de la fonction de coût f lorsque l'ensemble des noeuds visités et le dernier de ces noeuds sont fixés. Pour justifier N_2 , remarquons que, pour S l'ensemble des noeuds visités d'une route, si |S| est pair alors $|\bar{Q}(S)|$ est également pair. Le premier terme de la maximisation correspond au nombre de façons d'avoir un nombre impair d'origines visitées dont les destinations correspondantes ne le sont pas et le deuxième correspond au nombre de façons d'avoir un nombre pair de ces origines. En d'autres mots et en termes plus général, N_2 représente la combinatoire engendrée par les charges d_i associées aux demandes. N_3 s'explique par le fait que chaque ensemble, de type $\bar{Q}(S)$, du niveau 2 ne peut avoir plus de n noeuds terminaux différents. Et finalement N_4 et N_5 s'expliquent par la définition de T.

Donc $|\Sigma| \leq O(N_5) = O(N_2 \cdot n \cdot T)$ et la complexité de l'algorithme est de $O(N_2 \cdot n^2 \cdot T)$.

6.3 Construction et réduction du réseau

Nous voulons construire l'ensemble I des arcs du réseau, le plus petit possible. Nous construirons alors cet ensemble de façon à ne conserver que les arcs réalisables. De plus, nous associerons à chaque arc certaines informations permettant à l'algorithme des sous-problèmes d'être plus rapide.

6.3.1 Critères d'élimination des arcs

Les critères d'élimination sont basés sur les contraintes du problème et ils sont présentés au tableau 6.1. Les arcs qui ne satisferont jamais ces contraintes peuvent être éliminés. Notons que pour la contrainte de capacité, nous devons nous baser sur la plus grande capacité des véhicules de la

flotte utilisée, puisque ce réseau est commun à tous les sous-problèmes. Notons également qu'une nouvelle contrainte apparaît, celle de même lieu. Elle signifie que si nous devons servir 2 clients en même temps à un même lieu physique, sous certaines conditions (assurant l'optimalité), nous servons ces clients dans un ordre prédéterminé. Cette même contrainte a été utilisée dans Desrosiers, Soumis et Dumas [10] pour réduire la combinatoire des problèmes. Le tableau suivant présente l'élimination des arcs effectuée pour $i,j \in D$:

Tableau 6.1 : Critères d'élimination des arcs

contraintes	arcs éliminés	conditions	remarques
précédence	(i^-,i^+) et (g,i^-) $g \in G^+$	--	--
pairage	(i^+,g) $g \in G^-$	--	--
capacité	$(i^+,j^+),(i^+,j^-),(i^-,j^-)$	si $d_i + d_j > C^1$	C^1 est la plus grande capacité des véhicules.
horaire	(i,j)	si $a_i + t_{ij} + M_i > b_j$ ou si l'arc (i,j) est non post-réalisable dans le temps	Ce test peut-être renforcé lorsque tous les véhicules ont l'obligation de rejoindre le dépôt après leur tournée.
même lieu	$(i^+,j^+),(i^+,j^-),(i^-,j^-)$	l'arc (k,l) est éliminé si $t_{kl} = 0$ et (1) ou (2) est vérifié. (1) $b_k - M_l \geq b_l - M_k$ et $a_k \geq a_l$ avec au moins 1 inégalité stricte (2) $b_k - M_l = b_l - M_k$, $a_k = a_l$ et $k<l$.	lorsque $t_{kl}=0$ et sous certaines conditions un seul arc parmi (k,l) et (l,k) est suffisant pour assurer l'optimalité.

6.3.2 Informations conservées à chaque arc

Les informations supplémentaires que nous voulons conserver à chaque arc sont le "poids" (servant à l'élimination temporaire d'arcs à l'appel d'un sous-problème) et les temps de trajet (servant à faciliter la vérification de la post-réalisabilité).

Similairement au "saving" de Clarke & Wright, nous définissons un nouveau "saving" à chaque arc, comme étant le gain en déplacement obtenu en imbriquant 2 demandes étant donné que l'imbriquation doit se faire en passant par un arc spécifié. Notons que nous ne considérons pas le dépôt dans le calcul du "saving" et que ce calcul implique 4 noeuds au lieu de 2 comme dans C. & W.. Nous n'utilisons pas les "saving" directement mais une transformation des "saving": le "saving" s est transformé en α-s où α est une constante. Le "saving" transformé est appelé un **poids.**

Si un arc implique le passage obligatoire à d'autres noeuds (au plus 2), nous associons alors à l'arc les temps t' et t" de trajet correspondant aux 2 façons (s'il y a) de visiter ces noeuds additionnels.

Ces informations, pour chaque type d'arcs, sont résumées dans le tableau 6.2 pour $g \in V$ et $i,j \in D$ où

$D_{ij} = c_{i^+i^-}(d_i) + c_{j^+j^-}(d_j)$ est le coût de transport pour servir directement les demandes i et j séparément,

$f_{ij}(k,l)$ est le coût du plus court chemin satisfaisant les demandes i et j et passant par l'arc (k,l) et

α est une constante permettant de rendre les poids positifs.

Tableau 6.2 : Informations sur chaque arc

arcs	poids	t'	t"
(g^+,i^+)	0	0	0
(i^-,g^-)	0	0	0
(i^+,i^-)	α	0	0
(g^+,g^-)	0	0	0
(i^-,j^+)	$\alpha-[D_{ij}-f_{ij}(i^-,j^+)]$	$t_j{}^+{}_j{}^- + M_j{}^+$	idem
(i^+,j^-)	$\alpha-[D_{ij}-f_{ij}(i^+,j^-)]$	$t_j{}^-{}_i{}^- + M_j{}^-$	idem
(i^+,j^+)	$\alpha-[D_{ij}-f_{ij}(i^+,j^+)]$	$t_j{}^+{}_i{}^- + t_i{}^-{}_j{}^- + M_j{}^+ + M_i{}^-$	$t_j{}^+{}_j{}^- + t_j{}^-{}_i{}^- + M_j{}^+ + M_j{}^-$
(i^-,j^-)	$\alpha-[D_{ij}-f_{ij}(i^-,j^-)]$	0	0

Les arcs sont également triés par prédécesseur et par ordre croissant des poids.

L'élimination temporaire d'arcs se fait en éliminant les arcs dont les poids sont supérieurs à une certaine borne (voir la section 4.3 pour informations). Pour un arc (i^+,j^+) soit t' et t" les deux temps supplémentaires de déplacement associés alors la vérification de la post-réalisabilité se fait de la façon suivante:

soit $t_j{}^+$ l'heure d'arrivée au noeud j

si $t_j{}^+ + t' > b_i{}^-$ et si $t_j{}^+ + t'' > b_j{}^-$

alors la post réalisabilité ne sera pas satisfaite si nous utilisons l'arc (i^+,j^+).

CHAPITRE 7

RESULTATS

7.0 Résultats

Nous avons testé notre algorithme à partir de quatre problèmes réels fournis par GUINET [13] et de sept autres problèmes qui ont été générés de façon aléatoire. Nous avons également étudié la sensibilité des résultats et le comportement de l'algorithme face aux paramètres du problème tels: le nombre de demandes, le nombre de dépôts, les intervalles de temps et les quantités de marchandise transportées.

7.1 Description des problèmes

Chaque problème sera identifié par un code spécifiant quel groupe de demandes il traite et quelle flotte de véhicules il utilise. Par exemple: D1.V1 spécifie que le problème traite le groupe de demandes D1 avec la flotte de véhicules V1. Les deux prochaines sections porteront sur la description des demandes et des véhicules utilisés dans les applications. Dans ces applications la capacité des véhicules est identique (cette capacité est notée C^*).

7.1.1 Demandes

Le tableau 7.1 présente les principales caractéristiques de chaque groupe de demandes traitées. On retrouve: le nombre de demandes de transport d'un lieu à un autre, la répartition des demandes selon D_r (D_r est le nombre de demandes de charge $\leq 1/r \cdot C^*$), la quantité N_2 qui est une approximation de la combinatoire maximum engendrée par les charges associées aux demandes (définie à la section 6.2), la répartition des demandes selon leur temps de déplacement direct en minutes, la valeur maximum de ces temps et la largeur des intervalles de temps associés aux arrêts. Sauf avis contraire les demandes auront toutes des intervalles de temps non restrictifs aux arrêts leur étant associés.

Tableau 7.1 : Demandes de base

		dem. réelles		demandes aléatoires						
	groupe	D1	D2	D3	D4	D5	D6	D7	D8	D9
(1)	nb de dems	30	19	20	30	20	30	40	50	55
(2)	D_3	6	3	4	6	4	6	8	10	11
	D_2	18	11	12	18	12	18	24	30	33
	D_1	30	19	20	30	20	30	40	50	55
	N_2	154	56	67	154	67	154	277	436	529
(3)	t.d. [0,50)	1	.	3	2	.	.	7	6	4
	t.d.[50,100)	5	.	8	9	.	.	6	8	16
	t.d.[100,150)	10	.	7	7	.	.	12	15	15
	t.d.[150,200)	7	.	2	7	3	6	4	14	5
	t.d.[200,300)	6	5	.	5	4	7	11	7	15
	t.d.[300,400)	1	5	.	.	6	6	.	.	.
	t.d.[400,500)	.	4	.	.	4	5	.	.	.
	t.d.[500,600)	.	3	.	.	3	5	.	.	.
	t.d.[600,∞)	.	2	.	.	.	1	.	.	.
(4)	t.d. max	302	656	193	249	575	617	276	254	280
(5)	$b_i - a_i$, $i \in P$	•	•	•	•	•	•	•	•	•

(1) Nombre de demandes du groupe.

(2) $D_r \equiv$ Nombre de demandes avec charge $\leq (1/r) \cdot C^{\$}$.
N_2 évalue la combinatoire maximum engendrée par les quantités de marchandise à transporter.

(3) Répartition des demandes selon leur temps de déplacement (t.d.) en minutes.

(4) Temps de déplacement le plus grand.

(5) Largeur des intervalles de temps.

Les demandes D1 et D2 du tableau 7.1 sont des demandes issues de situations réelles vécues par une société de transport du sud-est de la France. Elles représentent l'activité journalière moyenne du transporteur de marchandise sur palettes, GUINET [13]. Une simple modification a été apportée à ces demandes, soit l'ajout d'un temps de manutention de la marchandise. Lorsque nous traitons des problèmes avec contraintes d'horaire, il devient essentiel de ne pas omettre ce

temps de manutention puisqu'il n'est pas négligeable.

A toutes les demandes, nous avons donc ajouté un temps de manutention à chaque arrêt comme suit:

$$\text{temps manutention} = 5 \text{ min.} + 25 \text{ min.} * \frac{\text{quantité de marchandise}}{\text{capacité du véhicule}}.$$

Pour un camion plein, nous établissons qu'il faut 25 minutes pour le décharger et nous ajoutons un temps fixe de 5 minutes pour le préparatif au déchargement.

Les charges des demandes sont assez près des quantités C^*, $1/2 \cdot C^*$ et $1/3 \cdot C^*$, ce qui implique que la quatité N_2 est une bonne approximation de la combinatoire engendrée par les charges. Notons également que des charges en volume et en poids sont utilisées mais que celles-ci ne sont pas distinguées, pour simplifier la présentation. Les quantités D_r ont été évaluées sur l'une ou l'autre (la plus représentative) des charges de poids ou volume.

La répartition des demandes permet de distinguer deux types de problèmes: un avec de petits déplacements (déplacements régionaux) et l'autre avec de longs déplacements (déplacements nationaux). On remarque alors que D1 est un groupe de demandes régionales et que D2 en est un de nationales (code souligné au tableau 7.1).

Les demandes des groupes D3 à D6 sont des demandes aléatoires générées à partir des 67 villes desservies par le transporteur cité plus haut. Nous avons généré deux types de groupes de demandes, des groupes de demandes régionales (D3 et D4) et des groupes de demandes nationales (D5 et D6).

Les demandes des groupes D7 à D9 sont des demandes également aléatoires mais par contre, elles sont issues d'un

réseau aléatoire de 200 villes. Seuls des groupes de demandes régionales ont été générés. Dans ces problèmes nous retrouvons moins le phénomène où plusieurs demandes peuvent avoir un arrêt en commun.

Pour les groupes de demandes D3 à D9, les charges associées aux demandes ont été choisies parmi l'ensemble $(C^*, 1/2 \cdot C^*, 1/3 \cdot C^*)$ et la répartition du nombre de demandes par charge a été fixée à 40 % pour C^*, 40 % pour $1/2 \cdot C^*$ et 20 % pour $1/3 \cdot C^*$. Cette répartition reflète à peu près la situation du groupe D1. Pour le groupe D2 elle est de 42 %, 42 % et 16 % respectivement. Des temps de manutention ont également été ajoutés de la même façon que présenté précédemment.

Pour permettre d'évaluer le comportement de l'algorithme lorsque les caractéristiques du problème changent, nous avons fait des extensions au groupe D1 (tableau 7.2). Les groupes D10 à D13 sont des extensions à D1 par l'ajout de fenêtres horaires (intervalles de temps restrictifs) aux arrêts de chaque demande. Afin d'avoir un horaire réalisable avec un nombre restreint de véhicules, les fenêtres horaires ont été générées à partir d'une solution heuristique au problème D1.V1 où la fenêtre horaire est définie autour de l'heure de passage du véhicule dans cette solution. De D10 à D13 nous accroissons la largeur de cette fenêtre autour de l'heure de passage; les demandes de D10 ont des intervalles fermés, ceux de D11 ont des intervalles d'une largeur 200 minutes (3 h 20), ceux de D12 de 400 minutes (6 h 40) et ceux de D13 de 600 minutes (10 h).

Les groupes D14 à D17 sont des extensions à D1 par la modification de la répartition des charges selon les quantités D_r (par conséquent N_2). La quantité N_2 étant fonction des quantités D_r, nous avons alors choisi ces dernières quantités de

façon à linéariser le plus possible la combinatoire engendrée par la répartition des charges, pour les groupes D14 à D17. Puisque N_2 maximise deux termes, et que l'un (le premier terme de la définition de la section 6.2) est constant à 30 pour D14 à D17 et que l'autre (le deuxième terme) a la séquence 1, 56, 106 et 154 pour D14 à D17, nous avons alors la séquence 30, 56, 106 et 154 pour N_2 (de D14 à D17). De D14 à D17 les problèmes deviennent de moins en moins restrictifs sur la capacité et de plus en plus difficiles à résoudre.

Tableau 7.2 : Variantes au groupe de demandes D1

		horaire				répartition des charges			
	groupe	D10	D11	D12	D13	D14	D15	D16	D17
(1)	D_3	6	6	6	6	0	0	0	0
	D_2	18	18	18	18	0	11	15	18
	D_1	30	30	30	30	30	30	30	30
	N_2	154	154	154	154	30	56	106	154
(2)	$b_i - a_i$, $i \in P$	0	200	400	600	•	•	•	•

(1) $D_r \equiv$ Nombre de demandes avec charge $\leq (1/r) \cdot C^k$.
N_2 évalue la combinatoire maximum engendrée par les quantités de marchandise à transporter.
(2) Largeur des intervalles de temps des arrêts associés aux demandes. Ces intervalles sont centrés sur l'heure de passage du véhicule d'une solution heuristique.

7.1.2 Véhicules

Le tableau 7.3 présente quelques caractéristiques des flottes de véhicules. Ces caractéristiques sont: le nombre de lieux d'origine et de destination différents des véhicules, le nombre de véhicules déjà utilisés (lieu de départ hors d'un dépôt), la vitesse de déplacement (kilomètres par heure: km/h) et les coûts de déplacement des véhicules à vide et avec charge

(litres par 100 km: l/km). Comme autres caractéristiques, il y a évidemment les intervalles de temps de disponibilité des véhicules et leur capacité. Comme mentionné au début du chapitre, tous les véhicules ont la même capacité C^*. Les intervalles de temps de disponibilité des véhicules ne sont pas présentés dans ce tableau puisqu'ils sont modifiées d'une application à l'autre, ils seront présentés dans les tableaux 7.4 à 7.6.

Nous avons considéré deux situations pour les lieux de destination. Dans un cas on oblige les véhicules d'une flotte à rejoindre leur dépôt respectif après leur tournée, et dans l'autre cas on ne l'oblige pas (noté par -- dans le tableau 7.3, indiquant qu'il n'y a pas de lieu de destination). Ces situations de véhicules déjà utilisés et de retour non obligatoire au dépôt surviennent lorsqu'un gros problème est décomposé en plusieurs périodes de temps et où chaque période consiste en la résolution d'un problème de taille réduite. Les lieux terminaux des véhicules à une période donnée deviennent les lieux d'origines des véhicules de la période suivante (Desrosiers, Soumis et Dumas [10]). Dans ces situations, pour faire en sorte d'utiliser en priorité les véhicules déjà sur la route, ceux-ci ont un coût fixe d'utilisation nul.

Pour les flottes V1 et V2, tous les véhicules partent d'un seul dépôt mais V1 diffère de V2 par le retour non obligatoire au dépôt de ses véhicules. Les flottes V2 à V6 se distinguent par le nombre de dépôts impliqués. Ils ont tous l'obligation de rejoindre leur dépôt respectif après leur tournée. Finalement les flottes V7 et V8 (provenant de situations réelles) n'ont pas l'obligation de rejoindre un dépôt après leur tournée et elles se distinguent par le nombre de véhicules déjà utilisés dans leur flotte. A ces dernières flottes ne sont associées qu'un seul dépôt.

Dans nos applications nous avons fait en sorte, qu'à chaque dépôt, le nombre de véhicules disponibles soit non restreignant.

Tableau 7.3 : Véhicules

	code	V1	V2	V3	V4	V5	V6	V7	V8
(1)	·nb de lieu d'origine	1	1	2	3	4	5	17	8
	·nb de lieu de destinat.	--	1	2	3	4	5	--	--
(2)	·nbre de véh. déja utilisés	0	0	0	0	0	0	22	16
(3)	·vitesse km/h	65	65	65	65	65	65	65	65
	·cout a vide l/km	32	32	32	32	32	32	32	32
	·cout avec charge l/km	44	44	44	44	44	44	44	44

(1) -- signifie que les véhicules n'ont pas l'obligation de rejoindre le dépôt après leur tournée.

(2) Le nombre de véhicules déjà utilisés correspond au nombre de véhicules dont le lieu d'origine n'est pas un dépôt.

(3) Vitesse et coûts de déplacement des véhicules. Le coût de déplacement étant différent selon qu'un véhicule est vide ou non.

7.2 Description des résultats

A partir des groupes de demandes et des flottes de véhicules présentés dans les sections précédentes, nous avons bâti et résolu 27 problèmes. Les résultats de la résolution de ces problèmes sont présentés dans les tableaux 7.4 à 7.6. Chaque résolution est décomposée en trois parties, soit une **DESCRIPTION** sommaire des caractéristiques du problème, les **RESULTATS** de la résolution (i.e. la valeur de la fonction objective et le temps de résolution) et une **ANALYSE** sommaire des **RESULTATS.**

Les principales caractéristiques des problèmes que l'on retrouve dans la partie **DESCRIPTION** sont : le nombre de demandes traitées (|D|), le nombre de types de véhicules (|V|, nombre de sous-problèmes générateurs), le nombre d'arcs du réseau (|I|, voir le chapitre 6 pour la construction de ce réseau), la largeur des intervalles de temps des chauffeurs (nombre d'heures de disponibilité), la largeur des intervalles de temps des arrêts associés aux demandes et finalement N2 qui évalue la combinatoire engendrée par les charges associées aux demandes. Ces caractéristiques sont les principaux facteurs qui influencent la résolution des problèmes et servent à évaluer leur difficulté.

Dans la partie **RESULTATS** nous dressons un bilan de la résolution de chaque problème. Nous retrouvons Z(pl) qui est la valeur de la fonction objective optimale du problème linéaire et Z(ple) qui est celle de la meilleure solution entière trouvée. Ces valeurs sont constituées du nombre de véhicules utilisés dont le lieu d'origine est un dépôt et du coût de déplacement des véhicules. La valeur Z(ple) n'est pas toujours la valeur optimale de la fonction objective du problème linéaire entier, mais celle d'une solution entière satisfaisante selon l'effort

fourni pour la trouver (plus le problème est difficile moins nous sommes exigeants). Nous donnons alors l'écart relatif entre Z(ple) et Z(pl) pour le nombre de véhicules utilisés et le coût de déplacement.

Nous retrouvons également dans cette partie, le temps d'exécution sur un CYBER 173, pour trouver Z(pl) et Z(ple). Notons que ce temps n'inclue pas la construction et la réduction du réseau tel que présenté dans la section 6.3.1. Ce temps est négligeable et est de l'ordre de 8 secondes pour le problème D1.V1.

Dans la partie **ANALYSE** des **RESULTATS** nous dressons les principaux facteurs justifiant les temps de résolution. Nous retrouvons le numéro du noeud dans l'arbre d'énumération où la meilleure solution entière a été trouvée, le nombre de noeuds explorés dans cet arbre, le nombre de demandes par route en moyenne (obtenu en divisant le nombre de demandes à traiter par le nombre de véhicules utilisés), le nombre de colonnes générées lors de la résolution et le nombre maximum d'états (S,j,t) conservés en même temps (i.e. la plus grande somme des cardinalités trouvée de Σ et Σ' (référer au chapitre 6)). Le nombre de noeuds explorés est l'indicateur de la complexité de la technique de SEP, le nombre de colonnes générées est celui de la technique de génération de colonnes et le nombre maximum d'états conservés est celui des sous-problèmes.

Tableau 7.4 : Problèmes de base

		problèmes réels				demandes aléatoires (réseau réel)				problèmes aléatoires		
	code	D1.V1	D1.V7	D2.V1	D2.V8	D5.V2	D6.V2	D3.V2	D4.V2	D7.V2	D8.V2	D9.V2
DESCRIPTION	nb de dem. (|D|)	30	30	19	19	20	30	20	30	40	50	55
	nb de sous-problèmes (|V|)	1	17	1	8	1	1	1	1	1	1	1
	nb d'arcs (|I|)	1176	1464	687	835	498	1077	534	1041	1299	1462	1996
	interv. chauffeurs ($b_g - a_g$, $g \in V$)	10 h	8 h	30 h	30 h	25 h	25 h	9 h	11 h	12 h	12 h	12 h
	interv. demandes ($b_i - a_i$, $i \in P$)	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
	N_2	154	154	56	56	67	154	67	154	277	436	529
RESULTATS	Z(pl)	10:1800	0:1755	6:3131	0:2894	7:3487	11:5461	8:1092	11:2121	15:3391	20:4271	22:4767
	Z(ple)	10:1811	0:1812	6:3131	0:2894	7:3487	11:5494	8:1100	11:2137	15:3426	20:4364	22:4918
	$\frac{Z(ple)-Z(pl)}{Z(pl)}$ %	0.0:0.6	0.0:3.2	0.0:0.0	0.0:0.0	0.0:0.0	0.0:0.6	0.0:0.7	0.0:0.7	0.0:1.0	0.0:2.2	0.0:3.2
	temps Z(pl)	112	256	92	89	27	112	28	111	66	95	204
	temps Z(ple)	142	827	95	96	29	139	51	169	172	215	313
ANALYSE RES	noeud de Z(ple)	2	4	1	2	1	10	2	9	2	8	42
	nb de noeuds explorés (ple)	6	25	1	2	1	10	7	9	10	22	43
	nb de demandes par routes (moy.)	3.0	2.0	3.2	1.7	2.9	2.7	2.5	2.7	2.7	2.5	2.5
	nb de col. géné.	115	241	99	82	61	131	86	135	139	155	220
	MAX (|Σ| + |Σ'|)	1326	487	1225	603	421	1366	326	1182	760	838	1840

où Z(pl) et Z(ple) sont respectivement les valeurs de la fonction objective du problème linéaire en nombres réels et en nombres entiers. MAX (|Σ| + |Σ'|) est le nombre maximum d'états conservés en même temps à une itération. N_2 est un indicatif sur la combinatoire engendré par la répar-des charges aux demandes et il est tel que défini dans la section 6.2.

Tableau 7.5 Variantes de D1.V1 (horaire)

		horaire (demandes)				D1.V1	horaire (chauffeurs)		
	code	D10.V1	D11.V1	D12.V1	D13.V1	D1.V1	D1.V1+1	D1.V1+2	D1.V1+3
DESCRIPTION	nb de dem. (|D|)	30	30	30	30	30	30	30	30
	nb de sous-problèmes (|V|)	1	1	1	1	1	1	1	1
	nb d'arcs (|I|)	1175	1175	1175	1175	1176	1384	1545	1653
	interv. chauffeurs ($b_g - a_g$, $g \in V$)	10 h	10 h	10 h	10 h	10 h	11 h	12 h	13 h
	interv. demandes ($b_i - a_i$, $i \in P$)	0 h	3 h 20	6 h 40	10 h	∞	∞	∞	∞
	N_2	154	154	154	154	154	154	154	154
RESULTATS	Z(pl)	13:2239	12:1965	10:1908	10:1843	10:1800	9:1780	8:1748	7:1721
	Z(ple)	13:2239	12:1981	10:1908	10:1843	10:1811	9:1792	8:1785	7:1734
	$\frac{Z(ple)-Z(pl)}{Z(pl)}$ %	0.0:0.0	0.0:0.8	0.0:0.0	0.0:0.0	0.0:0.6	0.0:0.7	0.0:2.1	0.0:0.8
	temps Z(pl)	7	30	44	92	112	148	237	323
	temps Z(ple)	13	66	50	97	142	258	690	603
ANALYSE RES	noeud de Z(ple)	1	7	1	1	2	3	3	2
	nb de noeuds explorés (ple)	1	7	1	1	6	7	21	7
	nb de demandes par routes (moy.)	2.3	2.5	3.0	3.0	3.0	3.3	3.75	4.3
	nb de col. géné.	54	110	92	102	115	133	223	174
	MAX (|Σ| + |Σ'|)	44	281	621	1170	1326	1921	2428	2774

où Z(pl) et Z(ple) sont respectivement les valeurs de la fonction objective du problème linéaire en nombres réels et en nombres entiers. MAX (|Σ| + |Σ'|) est le nombre maximum d'états conservés en même temps à une itération. N_2 est un indicatif sur la combinatoire engendré par la répartition des charges aux demandes et il est tel que défini dans la section 6.2.

Tableau 7.6 Variantes de D1.V1 (dépôt et charge)

		nombre de dépôts					restriction sur la charge			
	code	D1.V2	D1.V3	D1.V4	D1.V5	D1.V6	D14.V1	D15.V1	D16.V1	D17.V1
DESCRIPTION	nb de dem. (|D|)	30	30	30	30	30	30	30	30	30
	nb de sous-problèmes (|V|)	1	2	3	4	5	1	1	1	1
	nb d'arcs (|I|)	821	1593	1658	1725	1794	536	845	1075	1343
	interv. chauffeurs ($b_q - a_q$, $q \in V$)	10 h	10 h	10 h	10 h	10 h	10 h	10 h	10 h	10 h
	interv. demandes ($b_i - a_i$, $i \in P$)	∞	∞	∞	∞	∞	∞	∞	∞	∞
	N_2	154	154	154	154	154	30	56	106	154
RESULTATS	Z(pl)	12:2283	12:2267	12:2265	12:2264	12:2259	14:2248	12:2024	11:2064	11:1916
	Z(ple)	12:2283	12:2267	12:2265	12:2267	12:2272	14:2248	12:2074	11:2064	11:1932
	$\frac{Z(ple)-Z(pl)}{Z(pl)}$ %	0.0:0.0	0.0:0.0	0.0:0.0	0.0:0.1	0.0:0.6	0.0:0.0	0.0:2.5	0.0:0.0	0.0:0.8
	temps Z(pl)	47	87	107	124	154	14	36	85	173
	temps Z(ple)	51	92	113	130	212	17	101	88	181
ANALYSES	noeud de Z(ple)	1	1	1	1	2	1	29	1	1
	nb de noeuds explorés (ple)	1	1	1	2	6	1	29	1	4
	nb de demandes par routes (moy.)	2.5	2.5	2.5	2.5	2.5	2.1	2.5	2.7	2.7
	nb de col. géné.	92	116	126	125	150	86	151	95	111
	MAX (|E| + |E'|)	801	791	789	793	755	157	525	1198	2185

où Z(pl) et Z(ple) sont respectivement les valeurs de la fonction objective du problème linéaire en nombres réels et en nombres entiers. MAX (|E| + |E'|) est le nombre maximum d'états conservés en même temps à une itération. N_2 est un indicatif sur la combinatoire engendré par la répar- des charges aux demandes et il est tel que défini dans la section 6.2.

7.3 Analyse des résultats

L'analyse a pour but d'évaluer la performance de la méthode de résolution sur des problèmes "réels" et aléatoires et d'évaluer l'influence de différents facteurs, comme: le nombre de demandes, les intervalles de temps restrictifs sur les demandes, le nombre d'heures de disponibilité des chauffeurs, le nombre de dépôts (nombre de sous-problèmes) et finalement, la répartition de la quantité de marchandise à transporter pour les demandes (restriction sur la charge).

7.3.1 Performance globale

Des problèmes de 19 à 55 demandes (38 à 110 noeuds) et des problèmes de 30 demandes avec jusqu'à 17 sous-problèmes ont pu être résolus. Egalement, des problèmes jusqu'à 4.3 demandes (8.6 noeuds) en moyenne par route ont été résolus. Les tests nous ont alors montrés que 90 % du temps de résolution est attribué aux sous-problèmes.

Les solutions entières obtenues sont optimales pour le nombre de véhicules et le coût du déplacement se situe en moyenne à 0.8 % de l'optimum continu (3.2 % au maximum). Pour 11 des problèmes, l'optimum sur le déplacement a été prouvé. Pour les problèmes où l'écart entre Z(pl) et Z(ple) est grand, plusieurs noeuds ont dû être explorés dans l'arbre d'énumération et beaucoup de temps ordinateur a été nécessaire pour trouver une solution entière satisfaisante. Par exemple: D1.V7 à un écart de 3.2 % sur le déplacement par rapport à l'optimum continu et 25 noeuds de l'arbre de SEP ont dû être explorés.

7.3.2 Problèmes réels

Le tableau 7.4 présente 4 problèmes (D1.V1, D1.V7,

D2.V1 et D2.V8) issus de situations réelles vécues par un transporteur (Guinet [13]). Les problèmes utilisant le groupe de demande D1 correspondent à des problèmes régionaux (courts déplacements) et ceux utilisant le groupe de demande D2 correspondent à des problèmes nationaux (longs déplacements) (voir section 7.1.1).

Deux situations sont présentées: la première utilise la flotte de véhicule V1 et correspond à résoudre un problème réduit de première étape d'une décomposition alors que la seconde utilise la flotte de véhicule V7 ou V8 et correspond à résoudre un problème réduit d'étapes intermédiaires (voir section 7.1.2).

Les problèmes réduits d'étapes intermédiaires se distinguent des problèmes réduits de première étape par l'utilisation de véhicules hors d'un dépôt (avec coût fixe d'utilisation nul) et par le nombre de sous-problèmes (types de véhicules) à considérer. Le fait d'utiliser des véhicules à coût fixe nul rend le problème plus facile, puisqu'on n'a plus à minimiser le nombre de véhicules; par contre, le nombre élevé de sous-problèmes à traiter le rend plus difficile.

La comparaison des temps de résolution (112 versus 92 secs pour Z(pl) et 142 versus 95 secs pour Z(ple)) de D1.V1 avec D2.V1 nous indique que ce dernier est presque aussi difficilement résoluble que D1.V1 même s'il a moins de demandes à considérer (19 au lieu de 30), que son réseau possède moins d'arcs (687 au lieu de 1176), et que la combinatoire engendrée par les charges est plus faible (56 au lieu de 154). Ce qui rend D2.V1 aussi difficile à résoudre, c'est le grand intervalle de temps de disponibilité des chauffeurs (30 h au lieu de 10 h). En effet, la combinatoire du sous-problème est presque aussi grande

qu'en D1.V1 (MAX {|Σ| + |Σ'|}= 1225 au lieu de 1326) et le nombre de colonnes générées est semblable pour les deux problèmes (99 versus 115 colonnes).

La comparaison entre les problèmes de première étape et ceux d'étapes intermédiaires nous indique que ces derniers possèdent des sous-problèmes plus faciles à résoudre (MAX {|Σ| + |Σ'|}= 487 et 603 contre 1326 et 1225) mais qu'ils sont en plus grand nombre (8 et 17 sous-problèmes contre 1). Ce qui explique la plus grande facilité à les résoudre c'est la disponibilité de véhicules avec coût fixe nul de ces derniers. On remarque que le problème D2.V8 se résoud en un temps qui est semblable à celui de D2.V1 (96 et 95 secs) puisque l'avantage d'une résolution plus facile des sous-problèmes est atténué par leur plus grand nombre (8 sous-problèmes). Pour D1.V7, son nombre encore beaucoup plus élevé de sous-problèmes (au nombre de 17) fait qu'il consomme beaucoup plus de temps que D1.V1.

7.3.3 Problèmes à demandes aléatoires (réseau réel)

Le tableau 7.4 présente 4 problèmes (D5.V2, D6.V2, D3.V2 et D4.V2) dont les demandes sont aléatoires et créées à partir du réseau de villes du transporteur cité précédemment. Ces problèmes considèrent un seul dépôt et l'obligation de le rejoindre après leur tournée. Le temps de disponibilité des chauffeurs a été fixé entre 2.5 et 2.8 fois le temps de déplacement direct maximum des demandes de chaque problème.

Deux catégories de problèmes sont présentées: la première (D5.V2 et D6.V2) considère des demandes de longs déplacements (demandes nationales de durée entre 2 h 30 et 10 h) et la seconde (D3.V2 et D4.V2) des demandes de courts déplacements (demandes régionales de durée entre 0 h et 4 h)

(référer au tableau 7.1). Les deux catégories de problèmes réagissent sensiblement de la même façon.

On remarque que les deux problèmes de 20 demandes sont résolus très rapidement (29 et 51 secs) ce qui est expliqué par une faible combinatoire dans leur sous-problème (soit: 421 et 326 pour D5.V2 et D3.V2 respectivement). Les problèmes D6.V2 et D4.V2 sont résolus en des temps de 139 et 169 secs. respectivement.

7.3.4 Problèmes complètement aléatoires

Le tableau 7.4 présente 3 problèmes (D7.V2 à D9.V2) générés de façon aléatoire à partir de villes créées aléatoirement (voir section 7.1.1). Ces problèmes sont constitués de demandes de courts déplacements (durée inférieur à 4 h 40).

Comme pour les problèmes précédents, le temps de disponibilité des chauffeurs a été fixé entre 2.5 et 2.8 fois le temps de déplacement direct maximum des demandes, soit à 12 h.

Ces problèmes semblent plus faciles à résoudre que les problèmes issus du réseau de villes réelles.

Lorsque le nombre de demandes est augmenté, on remarque un croissance rapide du nombre d'arcs du réseau (1299 à 1996), de la combinatoire engendrée par les charges (277 à 529), du temps de résolution (172 à 313 secs), de la complexité de l'arbre de SEP (10 à 43 noeuds explorés), du nombre de colonnes générées (139 à 220) et de la combinatoire dans les sous-problèmes (760 à 1840 états maximum conservés). Les croissances semblent toutes exponentielles.

Remarquons que pour le problème D9.V2, il a fallu

explorer 43 noeuds pour trouver une solution entière optimale pour le nombre de véhicules mais à 3.2 % de l'optimum sur le coût de déplacement de la solution en nombres réels.

7.3.5 Problèmes avec horaire sur les demandes

Le tableau 7.5 présente 4 problèmes (D10.V1 à D13.V1), issus de D1.V1, avec des intervalles de temps de différentes largeurs aux arrêts associés aux demandes. Nous retrouvons également, dans le même tableau, un problème témoin D1.V1, ayant des intervalles de temps non restrictifs.

Comme discuté en 7.1.1, des intervalles fermés sont construits à chaque arrêt (problème D10) et sont progressivement élargis de chaque coté pour les problèmes D11 à D13. Pour le problème D11 les intervalles sont d'une largeur de 3 h 20, pour D12 elles sont d'une largeur de 6 h 40 et pour D13 elles sont d'une largeur de 10 h. Même si D13.V1 possède des intervalles aussi larges que la disponibilité du chauffeur, il ne correspond pas néanmoins au problème témoin puisque l'horaire des chauffeurs est fixé et que les intervalles sont élargis proportionnellement de chaque coté, n'assurant ainsi que la moitié supérieure des intervalles à être accessible par au moins un véhicule.

Les problèmes avec des intervalles de temps restrictifs sur les demandes sont très facilement résolus. Les effets d'accroître ces intervalles de temps sont: une diminution du nombre de véhicules utilisés (de 13 à 10 véhicules), une augmentation lente du temps de résolution (7 à 112 secs), une augmentation du nombre de colonnes générées (54 à 115 colonnes) et une augmentation de la combinatoire du sous-problème (44 à 1326). L'effet le plus bénéfique pour un transporteur est bien

entendu la diminution du nombre de véhicules utilisés, nous passons de 13 véhicules pour des intervalles fermés à 10 véhicules pour des intervalles élargis de 6h 40 min et plus. Remarquons que la croissance la plus marquée est la combinatoire du sous-problème.

Parmi les problèmes avec intervalles de temps, seul D11.V1 n'a pas donné une solution optimale entière et il a fallu explorer 7 noeuds avant d'obtenir une solution entière dont le coût de déplacement est à 0.8 % de celui de la solution en nombres réels.

A la vue de ces résultats, on peut facilement penser résoudre efficacement des problèmes de 75 et même 100 demandes (150 à 200 noeuds) pour lesquelles des intervalles de temps seront restrictifs.

7.3.6 Problèmes avec plus grande disponibilité des chauffeurs

Le tableau 7.5 présente 4 problèmes (D1.V1 à D1.V1+3) qui évaluent l'influence d'une augmentation des intervalles de temps de la disponibilité des chauffeurs. Les principaux effets en sont une augmentation du nombre d'arcs du réseau (presque linéaire, 1176 à 1653 arcs), une diminution du nombre de véhicules employés (linéaire, 10 à 7 véhicules), une augmentation du temps de résolution (exponentielle, 142 à 603 secs), une augmentation du nombre de colonnes générées (non monotone, 115 à 223 colonnes) et une croissance de la combinatoire du sous-problème (presque linéaire,1326 à 2774). Dû à la diminution de l'utilisation du nombre de véhicules, le nombre de demandes par routes en moyenne passe de 3.0 à 4.3 demandes (6.0 à 8.6 noeuds) par route.

Une certaine difficulté à trouver la meilleure solution entière est également apparente. En particulier pour le problème

D1.V1+2 où 21 noeuds ont dû être explorés et après le troisième noeud aucune autre meilleure solution entière ne fut trouvée.

7.3.7 Problèmes avec plusieurs dépôts

Le tableau 7.6 présente 5 problèmes (D1.V2 à D1.V6) dont le nombre de dépôts considérés augmente de 1 à 5.

Pour chacun d'eux, l'obligation que chaque véhicule rejoigne son dépôt respectif après sa tournée est exigée et un nombre de véhicules disponibles non contraignants à chaque dépôt est établi. L'effet direct de l'augmentation du nombre de dépôts est évidemment le nombre de sous-problèmes devant être traité (1 à 5 sous-problèmes).

Les autres effets principaux sont les suivants: une augmentation du nombre d'arcs (saut suivi d'une progression linéaire, 821 à 1794 arcs), une légère diminution de la fonction économique en nombres réels (12:2283 à 12:2259), une augmentation du temps de résolution (saut suivi d'une progression lente non linéaire, 51 à 212 secs) et une augmentation du nombre de colonnes générées (non monotone, 92 à 150 colonnes).

On remarque que la combinatoire de tous les sous-problèmes est semblable (Max $(|\Sigma| + |\Sigma'|)$ = 755 à 801). Alors la différence des temps de résolution des problèmes est due uniquement au nombre de colonnes générées.

7.3.8 Restriction sur la charge

Le tableau 7.6 présente 4 problèmes (D14.V1 à D17.V1) dont la répartition des charges associées aux demandes a été modifiée.

Le problème D14.V1 possède uniquement des demandes avec charges égales à la capacité du véhicule, ce qui implique que chaque demande est servie directement de son noeud d'origine à son noeud de destination (contraintes de capacité très restrictives). Il s'agit donc d'un problème de voyageur de commerce multiple avec une contrainte de durée sur les tournées. Pour les autres problèmes (D15.V1 à D17.V1) on augmente le nombre de demandes dont la charge est égale à la moitié de la capacité du véhicule (voir tableau 7.2) permetant ainsi l'imbriquation des demandes (contraintes de capacité moins restrictives). Cette augmentation est faite de façon à avoir une croissance linéaire de la combinatoire engendrée par les charges associées aux demandes, d'un problème à l'autre (voir la section 7.1.1 pour plus de détails).

Les principaux effets de cette croissance sont: une augmentation du nombre d'arcs (presque linéaire, 536 à 1343 arcs), une diminution considérable du nombre de véhicules utilisés (14 à 11 véhicules), une augmentation du temps de résolution (exponentielle, 17 à 181 secondes), une augmentation du nombre de colonnes générées (non monotone, 86 à 151 colonnes) et une augmentation de la combinatoire des sous-problèmes (exponentielle, MAX $\{|\Sigma| + |\Sigma'|\}$ = 157 à 2185).

On remarque que la restriction sur la charge a une grande influence sur les temps de résolution. Et finalement, que le problème D15.V1 se distingue des autres par une grande difficulté à trouver une solution entière qui soit à moins de 2.5 % de l'optimum du coût de déplacement de la solution optimale en nombres réels.

7.3.9 Synthèse des résultats

Les principales conclusions que nous pouvons tirer des sections précédentes sont les suivantes:

- A une croissance linéaire du nombre de demandes (|D|), correspond une croissance exponentielle du temps de résolution.

- A une croissance linéaire du nombre de dépôts (|V| > 1, types de véhicules), correspond une croissance lente du temps de résolution.

- A une croissance linéaire de la combinatoire engendrée par les charges associées aux demandes, correspond une croissance exponentielle du temps de résolution.

- Si l'objectif O1 (qui minimise uniquement le déplacement) du chapitre 1 est utilisé, alors les problèmes sont plus facilement résolus qu'avec l'objectif O2 (qui minimise le nombre de véhicules et le déplacement).

- Lorsque les intervalles de temps sont restrictifs, les problèmes se résolvent très facilement et on peut songer résoudre des problèmes de plus grande taille.

7.4 Comportements de l'algorithme

Dans la section 4.3, nous avons mentionné qu'il était préférable de résoudre le problème maître en plusieurs étapes. Les premières étapes étant des approximations du problème maître par l'utilisation de sous-ensembles d'arcs du réseau. Dans la présente section, nous étudions le comportement de l'algorithme d'étape en étape, la qualité des approximations effectuées et le comportement du sous-problème pour le problème de base D1.V1.

Pour toutes nos applications, nous avons défini des sous-ensembles d'arcs I" et I' de I de la façon suivante:

$$I'' = \{(i,j)\in I \mid P(i,j) \leq \bar{P}, \quad R(i,j) \leq 35\ \%\},$$

$$I' = \{(i,j)\in I \mid P(i,j) \leq \bar{P} + ECT(P)/2, \quad R(i,j) \leq 55\ \%\},$$

en exigeant de plus, un nombre minimum d'arcs sortant de chaque noeud, où $\bar{P}$ et $ECT(P)$ représentent respectivement la moyenne et l'écart-type des poids sur les arcs du réseau I.

Les étapes majeures de l'algorithme pour l'obtention d'une solution entière sont les suivantes:

1- Génération de colonnes en utilisant le sous-ensemble d'arcs I".
2- Génération de colonnes en utilisant le sous-ensemble d'arcs I'.
3- Génération de colonnes en utilisant l'ensemble d'arcs I.

   A la fin de la troisième étape nous obtenons une solution optimale du problème maître en nombres réels.

4- Addition d'une coupe sur le nombre de véhicules (voir la section 5.1) et génération de colonnes en utilisant le sous-ensemble d'arcs I".
5- Génération de colonnes en utilisant le sous-ensemble d'arcs I'.

6- Génération de colonnes en utilisant l'ensemble d'arcs I.

A la fin de la sixième étape nous obtenons une nouvelle solution optimale du problème maître en nombres réels avec un nombre entier de véhicules utilisés.

7 à...- Parcours de l'arbre de SEP où à chaque noeud (étape) on génère des colonnes en utilisant le sous-ensemble d'arcs I'.

A la fin de l'algorithme nous obtenons une solution entière optimale ou presqu'optimale.

Chaque étape majeure de l'algorithme consiste à appliquer le schéma de génération de colonnes présenté dans la section 3.1 en partant du problème maître restreint de l'étape précédente.

Afin de visualiser la répartition du temps de résolution et celle du nombre de colonnes générées à travers les étapes, nous avons confectionné deux graphiques (figures 7.1 et 7.2) qui présentent ces répartitions en fonction des étapes majeures de l'algorithme. De ces deux graphiques on remarque alors que la majorité des colonnes sont générées aux l'étapes majeures 1 et 4 (83 % des colonnes) et que peu de temps est consommé pendant ces étapes (42 % du temps total).

Figure 7.1 : Graphique du temps d'exécution vs les étapes

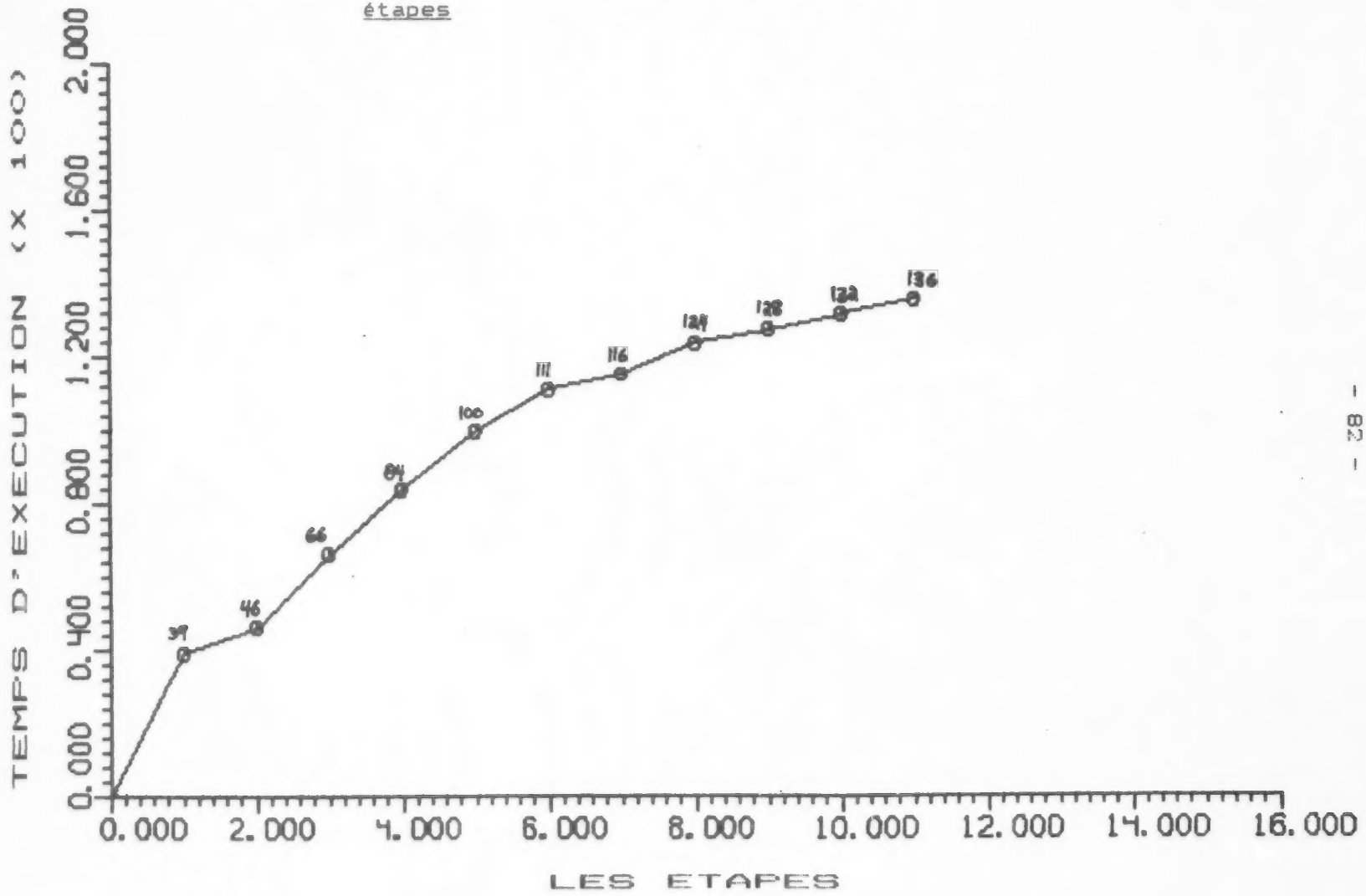
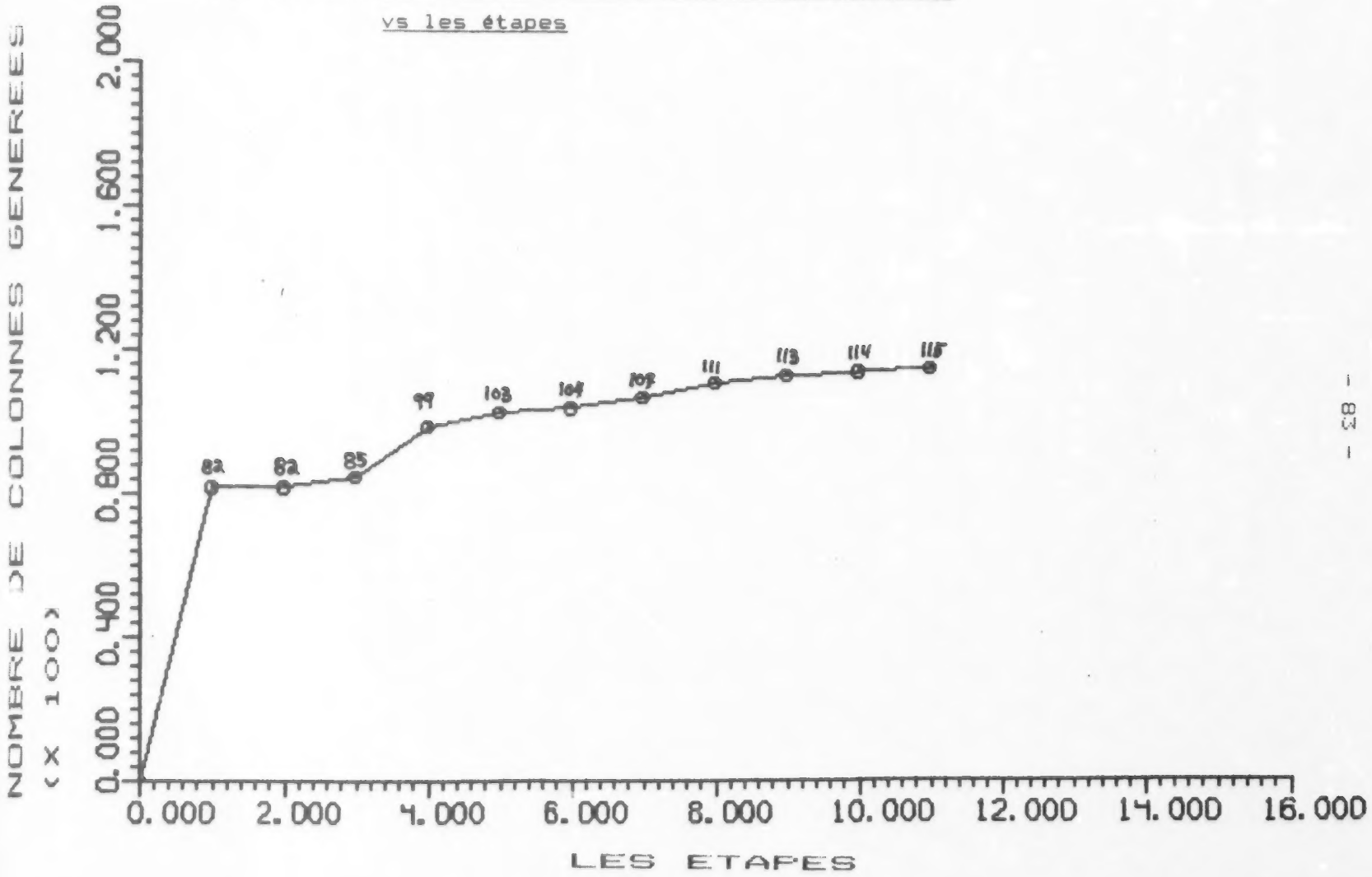


Figure 7.2 : Graphique du nombre de colonnes générées vs les étapes



Notons que les étapes 2, 3 et 6 n'ont pas permis de diminuer la fonction objective et que l'étape 5 n'a permis de la diminuer que de 0.2 % sur le coût de déplacement.

Le tableau 7.7 donne un autre aperçu de l'importance et l'utilité de chacune des étapes.

Tableau 7.7 : Compilation sur les étapes majeures

ETAPES MAJEURES	SECS/COL. GENEREES	TEMPS CONSOMME	NB COL. GENEREES	NB COL. UTILE(S) POUR Z(ple)
1	0.5	39	82	8
2	---	7	0	0
3	6.7	20	3	0
4	1.3	18	14	1
5	4.0	16	4	0
6	11.0	11	1	0
7 à 11	2.3	25	11	1

La deuxième colonne du tableau présente le temps moyen consommé par colonne générée; la troisième, le temps total consommé par étape(s); la quatrième, le nombre de colonnes générées par étape(s) et la dernière, le nombre de ces colonnes qui sont utilisées dans la meilleure solution entière trouvée. On remarque que les étapes 2, 3, 5 et 6 ont fourni peu de colonnes, que ces colonnes ont consommées beaucoup de temps et qu'elles n'ont pas été utiles pour la solution entière.

En éliminant les étapes 2, 3, 5 et 6 de l'algorithme on obtiendrait un nouvel algorithme de type approximatif (heuristique) pouvant fournir une solution optimale (c'est le cas pour D1.V1) et qui consommerait beaucoup moins de temps.

Maintenant, afin de visualiser le comportement du sous-problème, nous avons confectionné trois autres graphiques (voir les figures 7.3, 7.4 et 7.5) qui présentent le nombre de demandes considérées (référer à la section 4.2), le temps de résolution et le nombre d'états conservés, le tout en fonction du numéro d'ordre de la colonne générée.

Les trois graphiques présentent des formes similaires. On remarque les creux et les crêtes aux mêmes endroits mais avec des amplitudes différentes. On remarque également que le temps d'exécution est bien expliqué par le nombre de demandes considérées et le nombre maximum d'états conservés.

Figure 7.3 : Graphique du nombre de demandes traitées vs la colonne générée

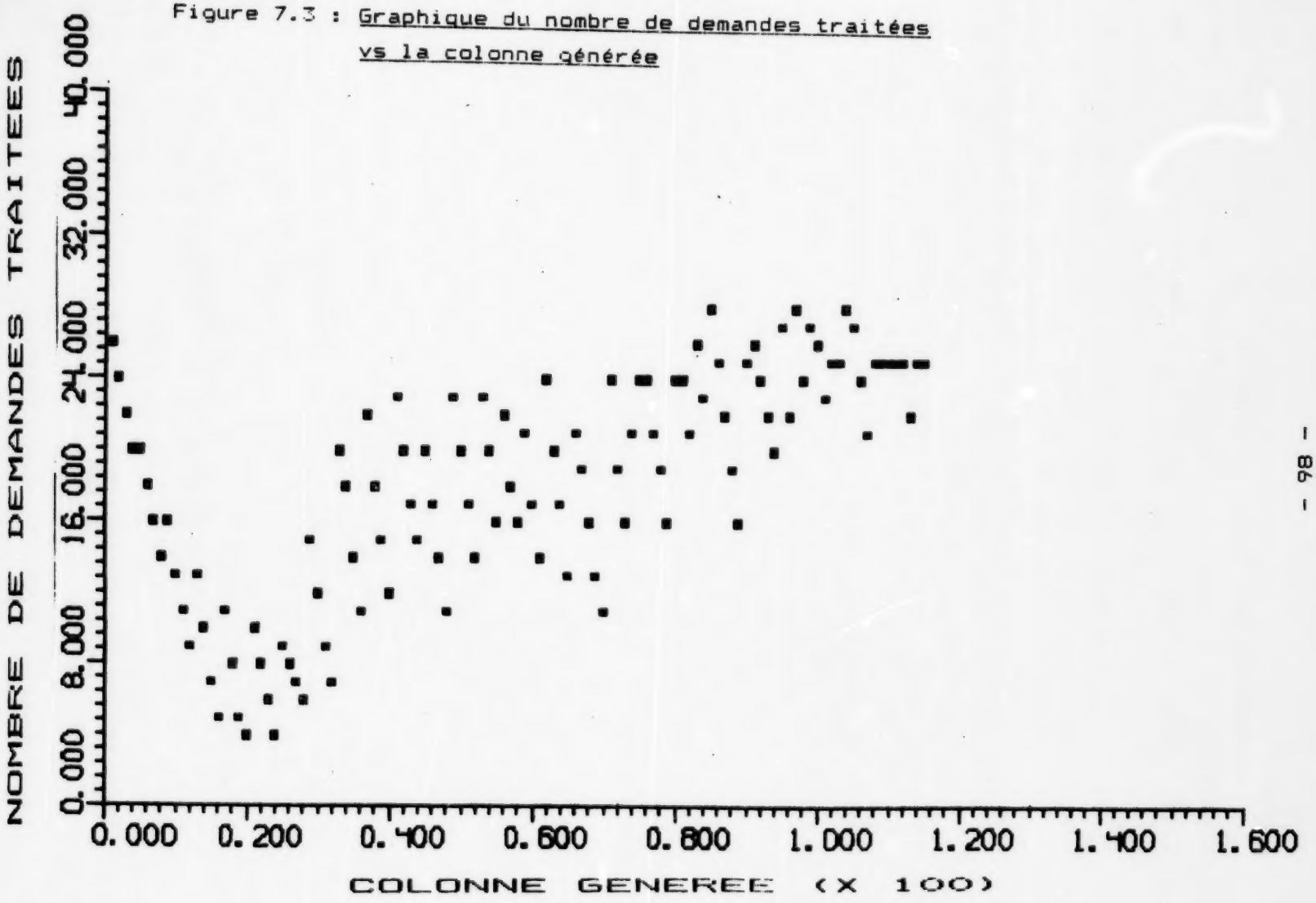


Figure 7.4 : Graphique du temps d'exécution vs la colonne générée

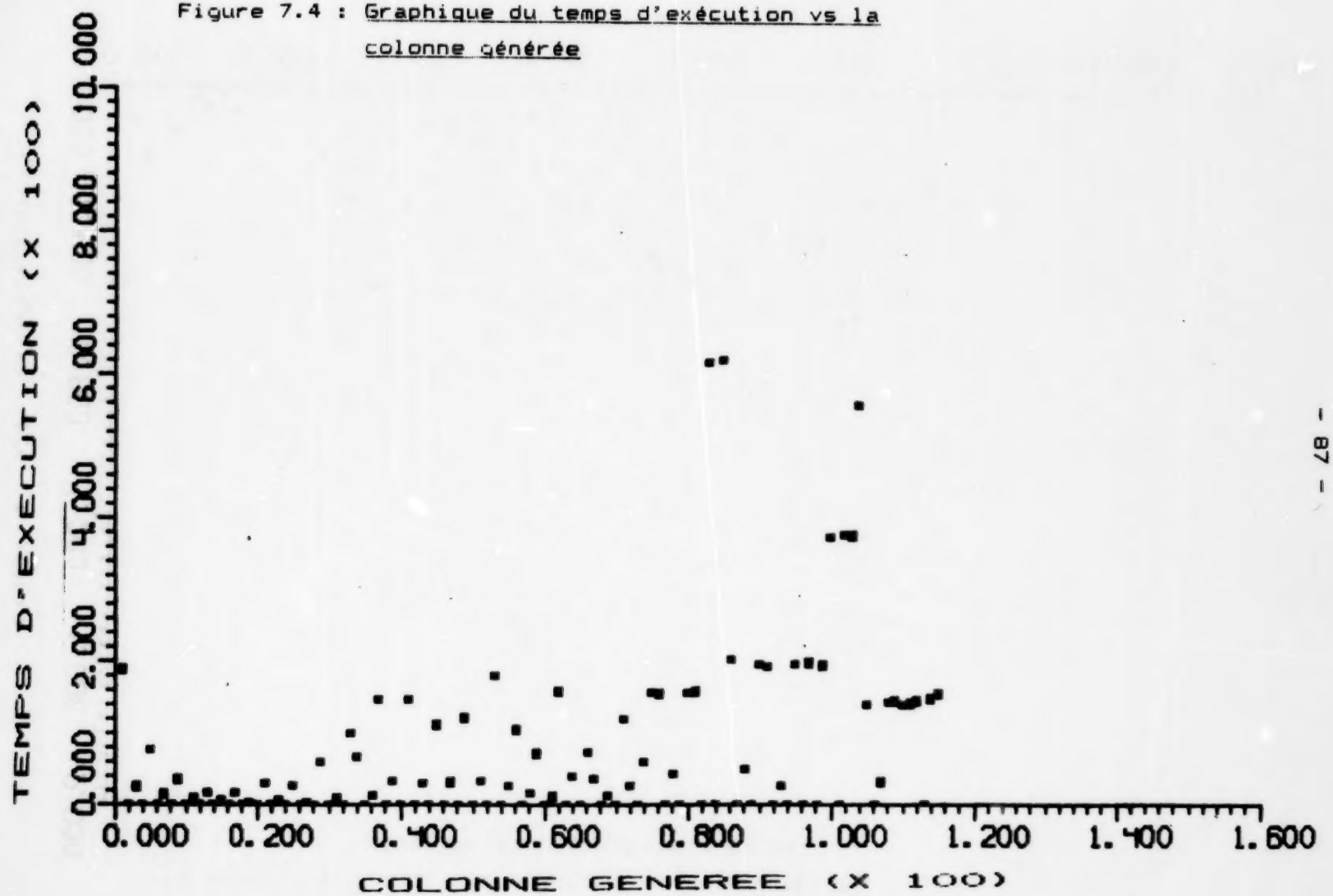
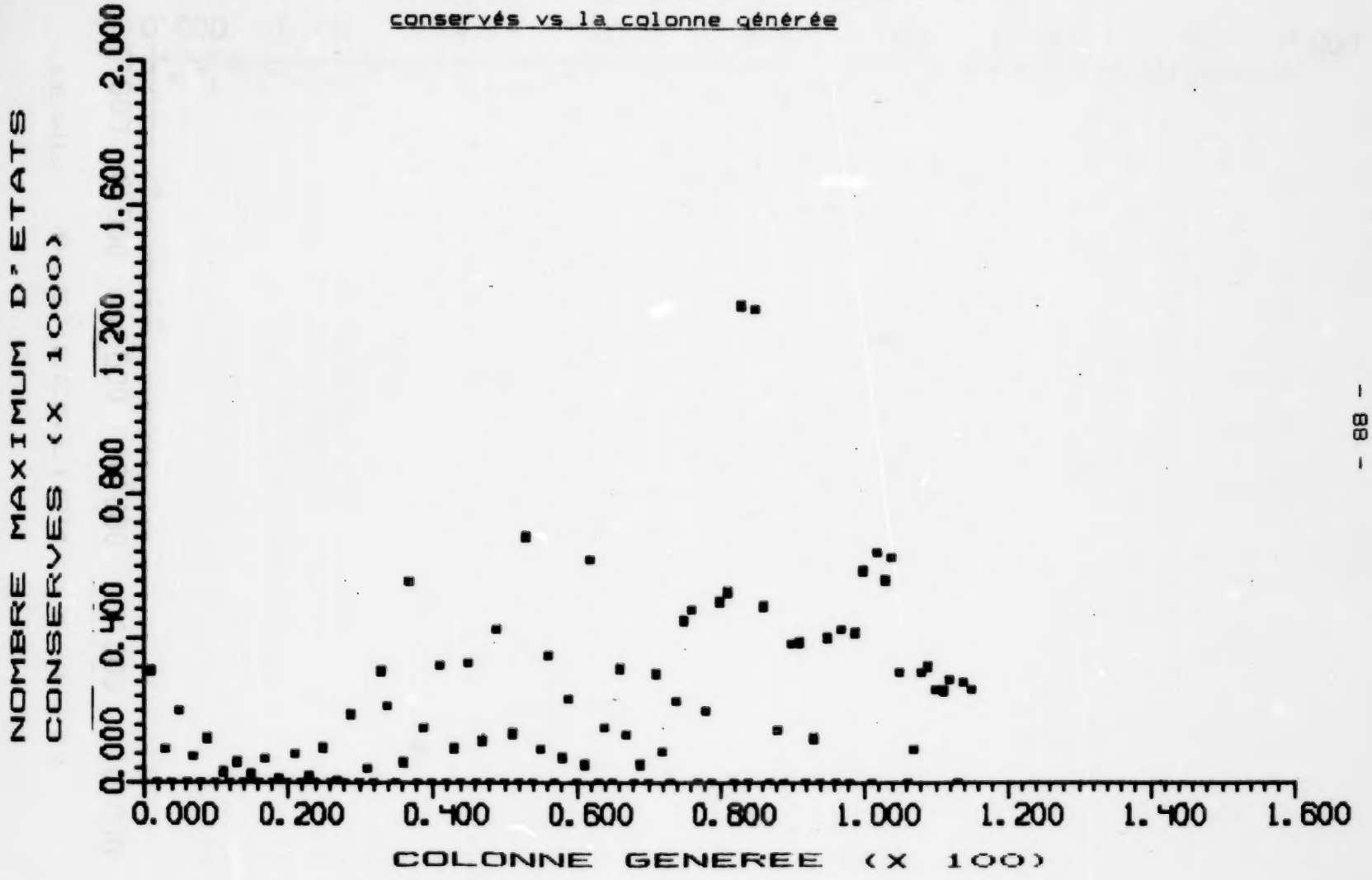


Figure 7.5 : Graphique du nombre maximum d'états conservés vs la colonne générée



7.5 Effets de différentes stratégies

Au chapitre 4, nous avons proposé plusieurs stratégies permettant d'accélérer la convergence de l'algorithme. Dans cette section, nous évaluons donc les stratégies les plus importantes. Le tableau 7.8 présente l'utilisation de trois stratégies soit la génération de colonnes multiples (stratégie 2) (section 4.2), la restriction sur les demandes considérées (statégie 4) (section 4.2) et la résolution par étapes avec l'utilisation de sous-ensemble d'arcs du réseau (stratégie 3) (section 4.3).

Tableau 7.8 : Comparaison des différentes stratégies pour Z(pl)

	D1.V1		D2.V1		D1.V7		D2.V8	
	absolue	relative	absolue	relative	absolue	relative	absolue	relative
(1) sans les stratégies 2,3 et 4	357	100 %	215	100 %	302	100 %	231	100 %
(2) plusieurs colonnes	202	57 %	142	66 %	361	120 %	175	76 %
(3) restriction de sur les arcs	248	69 %	164	76 %	202	67 %	146	63 %
(4) restriction sur les demandes	214	60 %	169	79 %	299	100 %	195	76 %
(5) stratégies (2)+(3)+(4)	112	31 %	92	43 %	256	85 %	89	39 %

(1) Le problème est résolu sans utiliser les stratégies (2) à (4).
(2) Plusieurs colonnes sont générées à chaque itération (voir la section 4.2) sans utiliser les stratégies (3) et (4).
(3) Le problème est résolu par étape en utilisant des sous-ensemble d'arcs de I (section 4.3) sans utiliser les stratégies (2) et (4).
(4) Le problème est résolu en appliquant la remarque (2) de la section 4.2 sans utiliser les stratégies (2) et (3).
(5) Le problème est résolu en utilisant les stratégies (2), (3) et (4).

Pour l'évaluation, nous considérons les quatres problèmes issus de situations réelles (D1.V1, D2.V1, D1.V7 et D2.V8). Rappelons que ces problèmes se regroupent en deux situations, soit celle où la flotte de véhicule V1 est utilisée (D1.V1 et D2.V1) et dont chaque véhicule a un coût fixe élevé d'utilisation et celle où la flotte de véhicules inclue des véhicules à coût fixe nul (D1.V7 et D2.V8). Le tableau 7.8 présente les comparaison en valeur relative (par rapport à la résolution sans les stratégies mentionnées) et en valeur absolue (en secondes CYBER 173).

Les problèmes de ces deux situations réagissent différemment selon les stratégies utilisées. Pour les problèmes D1.V1 et D2.V1 c'est la stratégie 2 qui individuellement est supérieure aux deux autres tandis que pour les problèmes D1.V7 et D2.V8 la stratégie 3 est supérieure. Remarquons qu'encore une fois, le problème D1.V7 se différencie des trois autres problèmes. Il s'en distingue par une piètre performance de résolution avec les stratégies 2 et 4 (valeur relative de résolution de 120 % et 100 % respectivement). Par contre, pour les autres problèmes toutes les stratégies donnent de très bons résultats.

Finalement, le regroupement des trois stratégies donne des résultats très intéressants pour tous les problèmes (avec des valeurs relatives de résolution de 31 %, 43 %, 85 % et 39 % pour D1.V1, D2.V1, D1.V7 et D2.V8 respectivement).

CHAPITRE 8

CONCLUSION

8. CONCLUSION

Nous avons dévelloppé une approche exacte pour résoudre le problème de confection de routes, d'une flotte mixte de véhicules, pour le transport de plusieurs origines à plusieurs destinations. La grande difficulté du problème n'a engendrée, à notre connaissance, que des méthodes heuristiques de résolution.

Lorsque les contraintes sont assez restrictives pour ne permettre que la visite de peu de clients (moins de 5 clients ou 10 noeuds) par route de véhicules, notre approche de recouvrement des demandes avec routes réalisables, résolue par génération de colonnes, obtient un vif succès. L'approche utilise des sous-problèmes de plus court chemin avec contraintes additionnelles où la majorité des contraintes s'y retrouvent et elle se prête bien à l'addition de nouvelles contraintes. De plus, la technique de génération de colonnes permet de pallier à la multitude de routes réalisables en ne générant qu'un sous-ensemble de ces routes. En somme, les techniques de relaxations et de restrictions appliquées aux sous-problèmes permettent de générer la majorité des colonnes intéressantes de façon à se raprocher rapidement de l'optimalité du problème maître. Ainsi, des problèmes jusqu'à 55 demandes (110 noeuds) et des problèmes de 30 demandes avec jusqu'à 17 types de véhicules ont pu être résolus à l'optimalité pour le nombre de véhicules et avec un écart moyen de 0.8 % sur le déplacement.

Puisque l'approche est primale, elle permet de dériver un algorithme heuristique pouvant résoudre de plus gros problèmes. Les suggestions apportées à la section 7.4, conduisent à une excellente méthode heuristique pouvant fournir une solution presque optimale et qui consomme beaucoup moins de temps. De plus cette méthode heuristique peut être utilisée dans un schéma de décomposition comme celui proposé par Desrosiers,

Soumis et Dumas [10] et discuté brièvement à la section 7.1.2, afin de résoudre des problèmes de très grandes tailles.

Etant donné le succès obtenu avec cette approche, nous croyons qu'il serait intéressant d'envisager la résolution de différents autres types de problèmes de transport avec une telle approche.

REMERCIEMENTS

Je tiens à remercier mon directeur de thèse, Jacques Desrosiers, de même que François Soumis pour les idées ainsi que les nombreux conseils apportés qui m'ont permis de mener à bien ce projet de recherche.

Je voudrais également témoigner ma reconnaissance envers Alain Guinet pour les jeux de données fournis sur des applications réelles et les membres de mon jury, Jacques Ferland et Michael Florian pour leurs critiques très constructives.

Je remercie finalement ma femme Michèle, pour sa patience et son encouragement apportés tout au long de cette recherche.

BIBLIOGRAPHIE

[1] BALINSKI, M., QUANDT, R.: "On an Integer Program for a Delevery Problem", Operations Research 12, 300-304 (1964).

[2] BELLMORE, M., MALONE, J.C.: "Pathology of Travelling Salesman Subtour Elimination Algorithm", Operations Research 19, 278-307 (1971).

[3] BODIN, L., GOLDEN, B.: "Classification in Vehicle Routing and Scheduling", Networks 11, 97-108 (1981).

[4] BODIN, L., GOLDEN, B., ASSAD, A., BALL, M.: "The Routing and Scheduling of Vehicles and Crews - The State of the Art", Computer and Operations Research, Special issue, (1983).

[5] CARPENETO, G., TOTH, P.: "Some New Branching and Bounding Criteria for the Asymetric Travelling Salesman Problem", Management Science 26, 736-743 (1980).

[6] CHRISTOFIDES, N.: "The Vehicle Routing Problem", RAIRO (recherche opérationnelle) 10, 55-70 (1976).

[7] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Mathematical Programming 20, 255-282 (1981).

[8] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "Space State Relaxation Procedure for the Computation of Bounds to Routing Problems", Networks 11, 145-164 (1981).

[9] DESROSIERS, J., SOUMIS, F., DESROCHERS, M.: "Routing with Time Windows by Column Generation", Networks 14, 545-565 (1984).

[10] DESROSIERS, J., SOUMIS, F., DUMAS, Y.: "The Multiples Vehicles Many to Many Routing Problem with Time Windows", Publication # 362, Centre de recherche sur les transport, Montréal, (1984).

[11] DESROSIERS, J., PELLETIER, P., SOUMIS, F.: "Plus Court Chemin avec Contraintes d'Horaires", RAIRO (recherche opérationnelle) 17, 357-377 (1983).

[12] GOLDEN, B., MAGNANTI, T.L., NGUYEN, H.Q.: "Implementing Vehicle Routing Algorithms", Networks 7, 113-148 (1977).

[13] GUINET, A.: "Le Système T.I.R.: un système d'établissement de tournées industrielles routières", Thèse de doctorat en informatique et automatique appliquées, Université Claude Bernard à Lyon, (1984).

[14] LAPORTE, G., NOBERT, Y.: "A Branch-and-Bound Algorithm for the Capacited Vehicle Routing Problem", OR Spectrum 5, 77-85 (1983).

[15] MAGNANTI, T.L.: "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects", Networks 11, 179-213 (1981).

[16] MILLER, C.E., TUCKER, A.W., ZEMLIN, R.A.: "Integer Programming Formulation of Travelling Salesman Problems", JACM 7, 326-329 (1969).

[17] NOBERT, Y.: "Construction d'algorithmes optimaux pour des extensions au Problème du Voyageur de Commerce", Publication # 297, Centre de recherche sur les transport, Montréal, (1982).

[18] RAO, M.R., ZIONTS, S.: "Allocation of Transportation Units to Alternative Trips - A Column Generation Scheme with Out-of-Kilter Subproblems", Operation Research 16, 52-63 (1968).

[19] SALKIN, H.M.: "Integer Programming", Addison-Wesley (1975).

[20] SOLOMON, M.: "Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms", College of Business Administration, Northeastern University, Working Paper 83-42 (1983).